1. Introduction. [LDF 2006.09.20.]

2. Copyright and licenses. [LDF 2006.10.23.]

The IWF Metadata Harvester is available for downloading from the following FTP server:
`ftp://ftp.gwdg.de/pub/gnu2/iwfmdh/`

Please send bug reports to `lfinsto1@gwdg.de`

The author can be contacted at:

Laurence D. Finston
Kreuzbergring 41
D-37075 Goettingen
Germany

Email: `lfinsto1@gwdg.de`
`s246794@stud.uni-goettingen.de`

3. Formatting commands. [LDF 2006.09.20.]
This section contains formatting commands. They don't appear in the TEX output of CWEAVE.

4. File Lists. [LDF 2006.09.18.]

5. Source Files. [LDF 2006.09.18.]

To Do

[LDF 2006.09.21.] Combine the files containing the code for the header file and the file for each **class**, where I haven't done this already.

---

6. Logical and Hierarchical Order. [LDF 2006.09.18.]

⟨ `resource.web` 22 ⟩	Preprocessor macro definitions for resources
⟨ `stdafx.web` 9 ⟩	Main header file for ZTest
⟨ `mainfrm.web` 25 ⟩	**class CMainFrame**
⟨ `ztestdoc.web` 47 ⟩	**class CZTestDoc**
⟨ `ztstview.web` 68 ⟩	**class CZTestView**
⟨ `ztest.web` 97 ⟩	**class CZTestApp** and **class CAboutDlg**
⟨ `dialogz1.web` 134 ⟩	**class Dialog_Z_1**
⟨ `scanner.web` 177 ⟩	The function *yylex*
⟨ `zclient.web` 188 ⟩	**class ZClient**
⟨ `ztstzoom.web` 364 ⟩	ZOOM functions
⟨ `opstrmtp.web` 384 ⟩	**struct Output_Stream_Type**
⟨ `picarcrd.web` 395 ⟩	**class Pica_Record**
⟨ `ctgcntnr.web` 418 ⟩	**class Category_Container**
⟨ `sbctgcnt.web` 565 ⟩	**class Subcategory_Container**
⟨ `dbcmmnd.web` 764 ⟩	**struct Database_Command**
⟨ `dbdspl.web` 778 ⟩	**class DB_Display**

7. ODBC Classes. [LDF 2006.09.27.]

⟨ picacats.web 998 ⟩	**class PICA_Categories**
⟨ picaflds.web 1019 ⟩	**class PICA_Fields**
⟨ pccatfld.web 1040 ⟩	**class PICA_Categories_PICA_Fields**
⟨ sources.web 1061 ⟩	**class Sources**
⟨ records.web 1082 ⟩	**class Records**
⟨ accnums.web 1103 ⟩	**class Access_Numbers**
⟨ callnums.web 1124 ⟩	**class Call_Numbers**
⟨ rccllnms.web 1145 ⟩	**class Records_Call_Numbers**
⟨ exprnums.web 1166 ⟩	**class Exemplar_Production_Numbers**
⟨ bibtyps.web 1208 ⟩	**class Bibliographic_Types**
⟨ bbtpcds.web 1187 ⟩	**class Bibliographic_Type_Codes**
⟨ rcbbtyps.web 1229 ⟩	**class Records_Bibliographic_Types**
⟨ physdesc.web 1250 ⟩	**class Physical_Descriptions**
⟨ rcphsdsc.web 1271 ⟩	**class Records_Physical_Descriptions**
⟨ authors.web 1292 ⟩	**class Authors**
⟨ recathrs.web 1313 ⟩	**class Records_Authors**
⟨ cntrbtrs.web 1334 ⟩	**class Contributors**
⟨ rccntrbt.web 1355 ⟩	**class Records_Contributors**
⟨ mnttls.web 1376 ⟩	**class Main_Titles**
⟨ rcmnttls.web 1397 ⟩	**class Records_Main_Titles**
⟨ contsums.web 1418 ⟩	**class Content_Summaries**
⟨ language.web 1439 ⟩	**class Languages**
⟨ reclang.web 1460 ⟩	**class Records_Languages**
⟨ subjtyps.web 1481 ⟩	**class Subject_Types**
⟨ subjects.web 1502 ⟩	**class Subjects**
⟨ recsubjs.web 1523 ⟩	**class Records_Subjects**
⟨ prmpttrn.web 1544 ⟩	**class Permutation_Patterns**
⟨ rmaccess.web 1565 ⟩	**class Remote_Access**
⟨ rcrmaccs.web 1586 ⟩	**class Records_Remote_Access**
⟨ publshrs.web 1607 ⟩	**class Publishers**
⟨ recpubs.web 1628 ⟩	**class Records_Publishers**
⟨ dbprovs.web 1649 ⟩	**class Database_Providers**
⟨ rcdbprov.web 1670 ⟩	**class Records_Database_Providers**
⟨ tempids.web 1691 ⟩	**class Temp_IDs**

8. Alphabetical Order. [LDF 2006.09.18.]

⟨ accnums.web 1103 ⟩	**class Access_Numbers**
⟨ authors.web 1292 ⟩	**class Authors**
⟨ bbtpcds.web 1187 ⟩	**class Bibliographic_Type_Codes**
⟨ bibtyps.web 1208 ⟩	**class Bibliographic_Types**
⟨ callnums.web 1124 ⟩	**class Call_Numbers**

⟨`contsums.web` 1418⟩	**class Content_Summaries**
⟨`ctgcntnr.web` 418⟩	**class Category_Container**
⟨`cntrbtrs.web` 1334⟩	**class Contributors**
⟨`dbcmmnd.web` 764⟩	**struct Database_Command**
⟨`dbdspl.web` 778⟩	**class DB_Display**
⟨`dbprovs.web` 1649⟩	**class Database_Providers**
⟨`dialogz1.web` 134⟩	**class Dialog_Z_1**
⟨`exprnums.web` 1166⟩	**class Exemplar_Production_Numbers**
⟨`language.web` 1439⟩	**class Languages**
⟨`mainfrm.web` 25⟩	**class CMainFrame**
⟨`mnttls.web` 1376⟩	**class Main_Titles**
⟨`opstrmtp.web` 384⟩	**struct Output_Stream_Type**
⟨`pccatfld.web` 1040⟩	**class PICA_Categories_PICA_Fields**
⟨`physdesc.web` 1250⟩	**class Physical_Descriptions**
⟨`picacats.web` 998⟩	**class PICA_Categories**
⟨`picaflds.web` 1019⟩	**class PICA_Fields**
⟨`picarcrd.web` 395⟩	**class Pica_Record**
⟨`prmpttrn.web` 1544⟩	**class Permutation_Patterns**
⟨`publshrs.web` 1607⟩	**class Publishers**
⟨`rcbbtyps.web` 1229⟩	**class Records_Bibliographic_Types**
⟨`rccllnms.web` 1145⟩	**class Records_Call_Numbers**
⟨`rccntrbt.web` 1355⟩	**class Records_Contributors**
⟨`rcdbprov.web` 1670⟩	**class Records_Database_Providers**
⟨`rcmnttls.web` 1397⟩	**class Records_Main_Titles**
⟨`rcphsdsc.web` 1271⟩	**class Records_Physical_Descriptions**
⟨`rcrmaccs.web` 1586⟩	**class Records_Remote_Access**
⟨`recathrs.web` 1313⟩	**class Records_Authors**
⟨`reclang.web` 1460⟩	**class Records_Languages**
⟨`records.web` 1082⟩	**class Records**
⟨`recpubs.web` 1628⟩	**class Records_Publishers**
⟨`recsubjs.web` 1523⟩	**class Records_Subjects**
⟨`resource.web` 22⟩	Preprocessor macro definitions for resources
⟨`rmaccess.web` 1565⟩	**class Remote_Access**
⟨`sbctgcnt.web` 565⟩	**class Subcategory_Container**
⟨`scanner.web` 177⟩	The function *yylex*
⟨`sources.web` 1061⟩	**class Sources**
⟨`stdafx.web` 9⟩	Main header file for ZTest
⟨`subjects.web` 1502⟩	**class Subjects**
⟨`subjtyps.web` 1481⟩	**class Subject_Types**
⟨`tempids.web` 1691⟩	**class Temp_IDs**
⟨`zclient.web` 188⟩	**class ZClient**
⟨`ztest.web` 97⟩	**class CZTestApp** and **class CAboutDlg**

⟨ `ztestdoc.web` 47 ⟩ .... **class CZTestDoc**
⟨ `ztstview.web` 68 ⟩ .... **class CZTestView**
⟨ `ztstzoom.web` 364 ⟩ .... ZOOM functions

9. stdafx (`stdafx.web`). [LDF 2006.10.05.]

Log

[LDF 2006.10.05.] Created this file. It contains code formerly in `stdafx.h` and `stdafx.cpp`.

⟨ `stdafx.web` 9 ⟩ ≡
  static char *id_string*[ ] = `"$Id:␣stdafx.web,v␣1.7␣2006/12/11␣10:15:59␣Administrator␣Exp␣$"`;

This code is cited in sections 6 and 8.

This code is used in section 20.

10. Preprocessor macro calls.

⟨ Preprocessor macro calls 10 ⟩ ≡
#pragma *once*

See also sections 27, 49, 70, 179, 190, 385, 396, 419, 566, 765, 779, 999, 1020, 1041, 1062, 1083, 1104, 1125, 1146, 1167, 1188, 1209, 1230, 1251, 1272, 1293, 1314, 1335, 1356, 1377, 1398, 1419, 1440, 1461, 1482, 1503, 1524, 1545, 1566, 1587, 1608, 1629, 1650, 1671, and 1692.

This code is used in sections 21, 46, 67, 96, 187, 363, 393, 416, 563, 762, 776, 995, 1017, 1038, 1059, 1080, 1101, 1122, 1143, 1164, 1185, 1206, 1227, 1248, 1269, 1290, 1311, 1332, 1353, 1374, 1395, 1416, 1437, 1458, 1479, 1500, 1521, 1542, 1563, 1584, 1605, 1626, 1647, 1668, 1689, and 1710.

11. Preprocessor macro definitions.

⟨ Preprocessor macro definitions 11 ⟩ ≡

```
#ifndef VC_EXTRALEAN
#define VC_EXTRALEAN
#endif
#ifndef WINVER
#define WINVER #0400
#endif
#ifndef _WIN32_WINNT
#define _WIN32_WINNT #0400
#endif
#ifndef _WIN32_WINDOWS
#define _WIN32_WINDOWS #0410
#endif
#ifndef _WIN32_IE
#define _WIN32_IE #0400
#endif
#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS
#define _AFX_ALL_WARNINGS
```

See also sections 28, 50, 71, and 99.

This code is used in sections 21, 45, 66, 95, and 132.

12. Include files in `stdafx.h`.

13. Windows and Visual Studio headers. [LDF Undated.]

⟨ Include files 13 ⟩ ≡
```
#include <afxwin.h>
#include <afxext.h>
#include <afxdisp.h>
#include <afxdtctl.h>
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>
#endif
#include <afxdb.h>
#include <atlenc.h>
```

See also sections 14, 15, 16, 17, 26, 48, 69, 98, 135, 178, 189, 365, 386, 397, 421, 568, 767, 781, 1001, 1022, 1043, 1064, 1085, 1106, 1127, 1148, 1169, 1190, 1211, 1232, 1253, 1274, 1295, 1316, 1337, 1358, 1379, 1400, 1421, 1442, 1463, 1484, 1505, 1526, 1547, 1568, 1589, 1610, 1631, 1652, 1673, and 1694.

This code is used in sections 21, 45, 66, 95, 132, 175, 186, 362, 382, 394, 417, 564, 763, 777, 996, 1018, 1039, 1060, 1081, 1102, 1123, 1144, 1165, 1186, 1207, 1228, 1249, 1270, 1291, 1312, 1333, 1354, 1375, 1396, 1417, 1438, 1459, 1480, 1501, 1522, 1543, 1564, 1585, 1606, 1627, 1648, 1669, 1690, and 1711.

14. C++ standard library headers. [LDF Undated.]

⟨ Include files 13 ⟩ +≡
```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <ios>
#include <iomanip>
#include <iostream>
#include <fstream>
#include <sstream>
#include <strstream>
#include <map>
#include <vector>
#include <algorithm>
#include <time.h>
#include <afxdb.h>
#include <afxmt.h>
#include <yaz/xmalloc.h>
#include <yaz/zoom.h>
```

15. Headers for types, variables, and constants defined in ZTest. [LDF 2006.09.06.]

Log

[LDF 2006.09.06.] Added this section with the # **include** command for `opstrmtp.h`, which contains the declaration of **struct Output_Stream_Type**.

[LDF 2006.10.23.] Now including `private.h`.

⟨ Include files 13 ⟩ +≡
```
#include "private.h"
#include "opstrmtp.h"
```

16. Headers for classes derived from **CRecordset**. [LDF 2006.07.24.]

Log

[LDF 2006.08.24.] Now including `mnttls.h` and `rcmnttls.h`. They contain the class declarations for **Main_Titles** and **Records_Main_Titles**, respectively.

[LDF 2006.08.24.] Now including `subjects.h` and `recsubjs.h`. They contain the class declarations for **Subjects** and **Records_Subjects**, respectively.

[LDF 2006.08.24.] Now including `subjtyps.h`. It contains the class declaration for **Subject_Types**.

[LDF 2006.08.24.] Now including `accnums.h`. It contains the class declaration for **Access_Numbers**.

[LDF 2006.08.24.] Now including `bibtyps.h` and `rcbbtyps.h`. They contain the class declarations for **Bibliographic_Types** and **Records_Bibliographic_Types**, respectively.

[LDF 2006.08.24.] Now including `callnums.h` and `rccllnms.h`. They contain the class declarations for **Call_Numbers** and **Records_Call_Numbers**, respectively.

[LDF 2006.08.24.] Now including `exprnums.h`. It contains the class declaration for **Exemplar_Production_Numbers**.

[LDF 2006.08.24.] Now including `prmpttrn.h`. It contains the class declaration for **Permutation_Patterns**.

[LDF 2006.08.25.] Now including `rmaccess.h`. It contains the class declaration for **Remote_Access**.

[LDF 2006.08.25.] Now including `rcrmaccs.h`. It contains the class declaration for **Records_Remote_Access**.

[LDF 2006.08.25.] Now including `picacats.h`. It contains the class declaration for **PICA_Categories**.

[LDF 2006.08.25.] Now including `picaflds.h`. It contains the class declaration for **PICA_Fields**.

[LDF 2006.08.25.] Now including `pccatfld.h`. It contains the class declaration for **PICA_Categories_PICA_Fields**.

[LDF 2006.08.25.] Now including `physdesc.h`. It contains the class declaration for **Physical_Descriptions**.

[LDF 2006.08.25.] Now including `rcphsdsc.h`. It contains the class declaration for **Records_Physical_Descriptions**.

[LDF 2006.08.25.] Now including `language.h`. It contains the class declaration for **Languages**.

[LDF 2006.08.25.] Now including `reclang.h`. It contains the class declaration for **Records_Languages**.

[LDF 2006.08.28.] Now including `publshrs.h`. It contains the class declaration for **Publishers**.

[LDF 2006.08.28.] Now including `recpubs.h`. It contains the class declaration for **Records_Publishers**.

[LDF 2006.08.28.] Now including `dbprovs.h`. It contains the class declaration for **Database_Providers**.

[LDF 2006.08.28.] Now including `rcdbprov.h`. It contains the class declaration for **Records_Database_Providers**.

[LDF 2006.09.06.] Now including `bbtpcds.h`. It contains the class declaration for **Bibliographic_Type_Codes**.

[LDF 2006.09.07.] Now including `contsums.h`. It contains the class declaration for **Content_Summaries**.

```
⟨Include files 13⟩ +≡
#include "Sources.h"    /* Sources */
#include "records.h"    /* Records */
#include "tempids.h"    /* Temp_IDs */
#include "Authors.h"    /* Authors */
#include "recathrs.h"    /* Records_Authors */
```

```
#include "cntrbtrs.h"    /* Contributors */
#include "rccntrbt.h"    /* Records_Contributors */
#include "mnttls.h"    /* Main_Titles */
#include "rcmnttls.h"    /* Records_Main_Titles */
#include "subjects.h"    /* Subjects */
#include "recsubjs.h"    /* Records_Subjects */
#include "subjtyps.h"    /* Subject_Types */
#include "accnums.h"    /* Access_Numbers */
#include "exprnums.h"    /* Exemplar_Production_Numbers */
#include "bbtpcds.h"    /* Bibliographic_Type_Codes */
#include "bibtyps.h"    /* Bibliographic_Types */
#include "rcbbtyps.h"    /* Records_Bibliographic_Types */
#include "callnums.h"    /* Call_Numbers */
#include "rccllnms.h"    /* Records_Call_Numbers */
#include "prmpttrn.h"    /* Permutation_Patterns */
#include "rmaccess.h"    /* Remote_Access */
#include "rcrmaccs.h"    /* Records_Remote_Access */
#include "picacats.h"    /* PICA_Categories */
#include "picaflds.h"    /* PICA_Fields */
#include "pccatfld.h"    /* PICA_Categories_PICA_Fields */
#include "physdesc.h"    /* Physical_Descriptions */
#include "rcphsdsc.h"    /* Records_Physical_Descriptions */
#include "language.h"    /* Languages */
#include "reclang.h"    /* Records_Languages */
#include "publshrs.h"    /* Publishers */
#include "recpubs.h"    /* Records_Publishers */
#include "dbprovs.h"    /* Database_Providers */
#include "rcdbprov.h"    /* Records_Database_Providers */
#include "contsums.h"    /* Content_Summaries */
```

17. Header files for types declared in ZTest. [LDF Undated.]

Log

[LDF 2006.07.20.] Now including `Subcategory_Container.h`.

[LDF 2006.08.23.] Now including `dbdspl.h`.

[LDF 2006.10.05.] Replaced `ldf_zoomtst` with `ztstzoom.h`.

```
⟨ Include files 13 ⟩ +≡
#include "ztstzoom.h"
#include "ZTest.h"
#include "dbcmmnd.h"    /* Database_Command */
#include "sbctgcnt.h"    /* Subcategory_Container */
#include "ctgcntnr.h"    /* Category_Container */
#include "picarcrd.h"    /* Pica_Record */
#include "ZClient.h"    /* ZClient */
#include "dbdspl.h"    /* DB_Display */
```

18. Global variables. [LDF 2006.08.23.]

Log

[LDF 2006.08.23.] Added this section with the **extern** declaration of **CMutex** *time_mutex*.

[LDF 2006.12.11.] Added the **extern** declaration of **string** *copyright_html_str*.

```
⟨ Global variables 18 ⟩ ≡
  extern CMutex time_mutex;
  extern string copyright_html_str;
```

See also section 127.

This code is used in sections 21 and 132.

19. Putting `stdafx.web` together.

20. This is what's compiled. [LDF Undated.]

```
#include "stdafx.h"
  ⟨ stdafx.web 9 ⟩
```

21. This is what's written to `stdafx.h`.

```
⟨ stdafx.h 21 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ Preprocessor macro definitions 11 ⟩
  ⟨ Include files 13 ⟩
  ⟨ Global variables 18 ⟩
```

22. Preprocessor macro definitions for resources (`resource.web`). [LDF 2006.10.12.]

Log

[LDF 2006.10.12.] Created this file. It contains code formerly in `resource.h`.

```
⟨ resource.web 22 ⟩ ≡    /* Empty section for use in cross-references. */
```

This code is cited in sections 6, 8, and 1717.

This code is used in section 1717.

23. Putting `resource.web` together.

24. This is what's written to `resource.h`.

```
⟨ resource.h 24 ⟩ ≡
  //{{NO_DEPENDENCIES}}      /* Microsoft Visual C++ generated include file. */
   /* Used by ZTest.rc */     /* */
#define IDD_ABOUTBOX 100
#define IDP_OLE_INIT_FAILED 100
#define IDR_MAINFRAME 128
#define IDR_ZTestTYPE 129
#define IDD_DIALOG1 131
#define IDC_SEARCH_COMMAND 1000
#define IDC_PERFORM_SEARCH 1002
#define IDC_PARSE_RECORDS 1003
#define IDC_DISPLAY_RECORDS 1004
#define IDC_WRITE_RECORDS_FILE 1005
#define IDC_CLEAR_DATABASE 1006
#define IDC_DISPLAY_ALL 1007
#define IDC_DISPLAY_RANGE 1008
#define IDC_DISPLAY_FROM_RECORD 1012
#define IDC_LIST4 1014
#define IDC_DISPLAY_TO_RECORD 1014      /* Next default values for new objects */
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE 132
#define _APS_NEXT_COMMAND_VALUE 32771
#define _APS_NEXT_CONTROL_VALUE 1015
#define _APS_NEXT_SYMED_VALUE 101
#endif
#endif
```

25. CMainFrame (`mainfrm.web`). [LDF 2006.10.12.]

Log

[LDF 2006.10.12.] Created this file. It contains code formerly in `MainFrm.h` and `MainFrm.cpp`.

```
⟨ mainfrm.web 25 ⟩ ≡
  static char id_string[] = "$Id:␣mainfrm.web,v␣1.5␣2006/10/23␣13:11:40␣Administrator␣Exp␣$";
```

This code is cited in sections 6 and 8.

This code is used in section 45.

26. Include files. [LDF Undated.]

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
#include "MainFrm.h"
```

27. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

28. Preprocessor macro definitions. [LDF Undated.]

```
⟨ Preprocessor macro definitions 11 ⟩ +≡
```

```
#ifdef _DEBUG
#define new  DEBUG_NEW
#endif
```

29. Declare class CMainFrame. [LDF Undated.]

```
⟨ Declare class CMainFrame 29 ⟩ ≡
  class CMainFrame : public CFrameWnd {
  public: ⟨ Declare public CMainFrame functions 34 ⟩
  protected: CStatusBar m_wndStatusBar;
    CToolBar m_wndToolBar;
    ⟨ Declare protected CMainFrame functions 32 ⟩
    DECLARE_MESSAGE_MAP( )
  };
```

This code is used in section 46.

30. Declare static indicators array. [LDF Undated.]

```
⟨ Declare static indicators array. 30 ⟩ ≡
  static UINT indicators[ ] = {ID_SEPARATOR, ID_INDICATOR_CAPS, ID_INDICATOR_NUM,
      ID_INDICATOR_SCRL, };
```

This code is used in section 45.

31. Functions. [LDF 2006.10.12.]

32. Constructor. [LDF Undated.]

```
⟨ Declare protected CMainFrame functions 32 ⟩ ≡
  CMainFrame(void);
  DECLARE_DYNCREATE(CMainFrame)
```

See also section 38.

This code is used in section 29.

33.

```
⟨ Define CMainFrame functions 33 ⟩ ≡
  CMainFrame::CMainFrame(void)
  {
    return;
  }
```

See also sections 35, 37, 39, 41, and 43.

This code is used in section 45.

34. Destructor. [LDF Undated.]

```
⟨ Declare public CMainFrame functions 34 ⟩ ≡
  virtual ~CMainFrame(void);
```

See also sections 36, 40, and 42.

This code is used in section 29.

35.

```
⟨ Define CMainFrame functions 33 ⟩ +≡
  CMainFrame :: ~CMainFrame(void)
  {
    return;
  }
```

36. Pre-Create Window. [LDF Undated.]

```
⟨ Declare public CMainFrame functions 34 ⟩ +≡
  virtual BOOL PreCreateWindow(CREATESTRUCT &cs);
```

37.

```
⟨ Define CMainFrame functions 33 ⟩ +≡
  BOOL CMainFrame :: PreCreateWindow(CREATESTRUCT &cs)
  {
    if (¬CFrameWnd :: PreCreateWindow(cs)) return FALSE;
    return TRUE;
  }
```

38. On Create. . [LDF Undated.]

```
⟨ Declare protected CMainFrame functions 32 ⟩ +≡
  afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
```

39.

```
⟨ Define CMainFrame functions 33 ⟩ +≡
  int CMainFrame :: OnCreate(LPCREATESTRUCT lpCreateStruct)
  {
    if (CFrameWnd :: OnCreate(lpCreateStruct) ≡ −1) return −1;
    if (¬m_wndToolBar.CreateEx(this,
          TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE | CBRS_TOP | CBRS_GRIPPER | CBRS_TOOLTIPS |
          CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ∨ ¬m_wndToolBar.LoadToolBar(IDR_MAINFRAME)) {
      TRACE0("Symbolleiste␣konnte␣nicht␣erstellt␣werden\n");
      return −1;    /* Fehler bei Erstellung */
    }
    if (¬m_wndStatusBar.Create(this) ∨ ¬m_wndStatusBar.SetIndicators(indicators, sizeof
          (indicators)/sizeof(UINT))) {
      TRACE0("Statusleiste␣konnte␣nicht␣erstellt␣werden\n");
      return −1;    /* Fehler bei Erstellung */
    }
    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);
    return 0;
  }    /* End of CMainFrame :: OnCreate definition. */
```

40. Assert Valid. [LDF Undated.]

```
⟨ Declare public CMainFrame functions 34 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid(void) const;
#endif
```

41.

```
⟨ Define CMainFrame functions 33 ⟩ +≡
#ifdef _DEBUG
  void CMainFrame::AssertValid(void) const
  {
    CFrameWnd::AssertValid();
  }
#endif
```

42. Dump. [LDF Undated.]

```
⟨ Declare public CMainFrame functions 34 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

43.

```
⟨ Define CMainFrame functions 33 ⟩ +≡
#ifdef _DEBUG
  void CMainFrame::Dump(CDumpContext &dc) const
  {
    CFrameWnd::Dump(dc);
  }
#endif
```

44. Putting CMainFrame together.

45. This is what's compiled.

```
  ⟨ Include files 13 ⟩
  ⟨ mainfrm.web 25 ⟩
  ⟨ Preprocessor macro definitions 11 ⟩
  IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)
  BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
  ON_WM_CREATE()
  END_MESSAGE_MAP()
  ⟨ Declare static indicators array. 30 ⟩
  ⟨ Define CMainFrame functions 33 ⟩
```

46. This is what's written to `mainfrm.h`.

⟨ `mainfrm.h` 46 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ Declare **class CMainFrame** 29 ⟩

47. CZTestDoc (`ztestdoc.web`). [LDF 2006.10.11.]

Log

[LDF 2006.10.11.] Created this file. It contains code formerly in `ZTestDoc.h` and `ZTestDoc.cpp`.

⟨ `ztestdoc.web` 47 ⟩ ≡
  static char *id_string*[ ] = `"$Id:␣ztestdoc.web,v␣1.4␣2006/10/23␣13:00:20␣Administrator␣Exp␣$"`;

This code is cited in sections 6 and 8.

This code is used in section 66.

48. Include files.

⟨ Include files 13 ⟩ +≡
#include `"stdafx.h"`
#include `"ztestdoc.h"`

49. Preprocessor macro calls.

⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma *once*

50. Preprocessor macro definitions.

⟨ Preprocessor macro definitions 11 ⟩ +≡
#ifdef `_DEBUG`
#define new `DEBUG_NEW`
#endif

51. Declare class CZTestDoc. [LDF Undated.]

⟨ Declare **class CZTestDoc** 51 ⟩ ≡
  class CZTestDoc : **public CDocument** {
  protected: ⟨ Declare **protected CZTestDoc** functions 53 ⟩
  public: ⟨ Declare **public CZTestDoc** functions 55 ⟩
  protected: `DECLARE_MESSAGE_MAP`( )
  };

This code is used in section 67.

52. Functions. [LDF 2006.10.11.]

53. Constructor.

⟨ Declare **protected CZTestDoc** functions 53 ⟩ ≡
  CZTestDoc(**void**); `DECLARE_DYNCREATE`(**CZTestDoc**)

This code is used in section 51.

54.

⟨ Define **CZTestDoc** functions 54 ⟩ ≡
```
  CZTestDoc :: CZTestDoc( )
  {
    return;
  }
```
See also sections 56, 58, 60, 62, and 64.

This code is used in section 66.

55. Destructor.

⟨ Declare **public CZTestDoc** functions 55 ⟩ ≡
```
  virtual ~CZTestDoc(void);
```
See also sections 57, 59, 61, and 63.

This code is used in section 51.

56.

⟨ Define **CZTestDoc** functions 54 ⟩ +≡
```
  CZTestDoc :: ~CZTestDoc( )
  {
    return;
  }
```

57. On New Document. [LDF Undated.]

⟨ Declare **public CZTestDoc** functions 55 ⟩ +≡
```
  virtual BOOL OnNewDocument( );
```

58.

⟨ Define **CZTestDoc** functions 54 ⟩ +≡
```
  BOOL CZTestDoc :: OnNewDocument( )
  {
    if (¬CDocument :: OnNewDocument( )) return FALSE;
    return TRUE;
  }
```

59. Serialize. [LDF Undated.]

⟨ Declare **public CZTestDoc** functions 55 ⟩ +≡
```
  virtual void Serialize(CArchive &ar);
```

60.

```
⟨ Define CZTestDoc functions 54 ⟩ +≡
  void CZTestDoc::Serialize(CArchive &ar)
  {
    if (ar.IsStoring()) {     /* TODO: Hier Code zum Speichern einfügen */
    }
    else {     /* TODO: Hier Code zum Laden einfügen */
    }
  }
```

61. Assert Valid. [LDF Undated.]

```
⟨ Declare public CZTestDoc functions 55 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

62.

```
⟨ Define CZTestDoc functions 54 ⟩ +≡
#ifdef _DEBUG
  void CZTestDoc::AssertValid( ) const
  {
    CDocument::AssertValid( );
  }
#endif
```

63. Dump. [LDF Undated.]

```
⟨ Declare public CZTestDoc functions 55 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

64.

```
⟨ Define CZTestDoc functions 54 ⟩ +≡
#ifdef _DEBUG
  void CZTestDoc::Dump(CDumpContext &dc) const
  {
    CDocument::Dump(dc);
  }
#endif
```

65. Putting CZTestDoc together.

66. This is what's compiled.

```
  ⟨ Include files 13 ⟩
  ⟨ ztestdoc.web 47 ⟩
  IMPLEMENT_DYNCREATE(CZTestDoc, CDocument)
  BEGIN_MESSAGE_MAP(CZTestDoc, CDocument)
  END_MESSAGE_MAP( )
  ⟨ Preprocessor macro definitions 11 ⟩
  ⟨ Define CZTestDoc functions 54 ⟩
```

67. This is what's written to `ztestdoc.h`.

```
⟨ ztestdoc.h 67 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ Declare class CZTestDoc 51 ⟩
```

68. CZTestView (`ztstview.web`). [LDF 2006.10.11.]

Log

[LDF 2006.10.11.] Created this file. It contains code formerly in `ZTestView.h` and `ZTestView.cpp`.

```
⟨ ztstview.web 68 ⟩ ≡
  static char id_string[] = "$Id:␣ztstview.web,v␣1.4␣2006/10/23␣13:03:31␣Administrator␣Exp␣$";
```

This code is cited in sections 6 and 8.

This code is used in section 95.

69. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
#include "ZTestDoc.h"
#include "ztstview.h"
```

70. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

71. Preprocessor macro definitions.

```
⟨ Preprocessor macro definitions 11 ⟩ +≡
#ifdef _DEBUG
#define new  DEBUG_NEW
#endif
```

72. Declare class CZTestView. [LDF Undated.]

```
⟨ Declare class CZTestView 72 ⟩ ≡
  class CZTestView : public CView {
  protected: ⟨ Declare protected CZTestView functions 74 ⟩
  public: ⟨ Declare public CZTestView functions 76 ⟩
  protected: DECLARE_MESSAGE_MAP( )
  };
```

This code is used in section 96.

73. Functions. [LDF 2006.10.11.]

74. Constructor.

```
⟨ Declare protected CZTestView functions 74 ⟩ ≡
  CZTestView( ); DECLARE_DYNCREATE(CZTestView)
```

See also sections 84, 86, and 88.

This code is used in section 72.

75.

```
⟨ Define CZTestView functions 75 ⟩ ≡
  CZTestView :: CZTestView(void)
  {
    return;
  }
```

See also sections 77, 79, 81, 83, 85, 87, 89, 91, and 93.

This code is used in section 95.

76. Destructor. [LDF Undated.]

```
⟨ Declare public CZTestView functions 76 ⟩ ≡
  virtual ~CZTestView(void);
```

See also sections 78, 80, 82, 90, and 92.

This code is used in section 72.

77.

```
⟨ Define CZTestView functions 75 ⟩ +≡
  CZTestView :: ~CZTestView(void)
  {
    return;
  }
```

78. Get Document. [LDF Undated.]

```
⟨ Declare public CZTestView functions 76 ⟩ +≡
  CZTestDoc *GetDocument(void) const;
```

79.

```
⟨ Define CZTestView functions 75 ⟩ +≡
#ifndef _DEBUG
  inline CZTestDoc *CZTestView :: GetDocument(void) const
  {
    return reinterpret_cast⟨CZTestDoc *⟩(m_pDocument);
  }
#endif
#ifdef _DEBUG
  CZTestDoc *CZTestView :: GetDocument( ) const      /* Nicht-Debugversion ist inline */
  {
    ASSERT(m_pDocument→IsKindOf(RUNTIME_CLASS(CZTestDoc)));
    return (CZTestDoc *) m_pDocument;
  }
#endif
```

80. On Draw. [LDF Undated.]

```
⟨ Declare public CZTestView functions 76 ⟩ +≡
  virtual void OnDraw(CDC *pDC);
```

81.

```
⟨ Define CZTestView functions 75 ⟩ +≡
  void CZTestView::OnDraw(CDC *pDC)
  {
    CZTestDoc *pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    if (¬pDoc) return;
  }
```

82. Pre-Create Window. . [LDF Undated.]

```
⟨ Declare public CZTestView functions 76 ⟩ +≡
  virtual BOOL PreCreateWindow(CREATESTRUCT &cs);
```

83.

```
⟨ Define CZTestView functions 75 ⟩ +≡
  BOOL CZTestView::PreCreateWindow(CREATESTRUCT &cs)
  {
    return CView::PreCreateWindow(cs);
  }
```

84. On Prepare Printing. [LDF Undated.]

```
⟨ Declare protected CZTestView functions 74 ⟩ +≡
  virtual BOOL OnPreparePrinting(CPrintInfo *pInfo);
```

85.

```
⟨ Define CZTestView functions 75 ⟩ +≡
  BOOL CZTestView::OnPreparePrinting(CPrintInfo *pInfo)
  {
    return DoPreparePrinting(pInfo);
  }
```

86. On Begin Printing. [LDF Undated.]

```
⟨ Declare protected CZTestView functions 74 ⟩ +≡
  virtual void OnBeginPrinting(CDC *pDC, CPrintInfo *pInfo);
```

87.

```
⟨ Define CZTestView functions 75 ⟩ +≡
  void CZTestView::OnBeginPrinting(CDC *pDC, CPrintInfo *pInfo)
  {
    return;
  }
```

88. On End Printing. [LDF Undated.]

```
⟨ Declare protected CZTestView functions 74 ⟩ +≡
  virtual void OnEndPrinting(CDC *pDC, CPrintInfo *pInfo);
```

89.

```
⟨ Define CZTestView functions 75 ⟩ +≡
  void CZTestView :: OnEndPrinting(CDC *pDC, CPrintInfo *pInfo)
  {
    return;
  }
```

90. Assert Valid. [LDF Undated.]

```
⟨ Declare public CZTestView functions 76 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

91.

```
⟨ Define CZTestView functions 75 ⟩ +≡
#ifdef _DEBUG
  void CZTestView :: AssertValid( ) const
  {
    CView :: AssertValid( );
  }
#endif
```

92. Dump. [LDF Undated.]

```
⟨ Declare public CZTestView functions 76 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

93.

```
⟨ Define CZTestView functions 75 ⟩ +≡
#ifdef _DEBUG
  void CZTestView :: Dump(CDumpContext &dc) const
  {
    CView :: Dump(dc);
  }
#endif
```

94. Putting CZTestView together.

95. This is what's compiled.

```
  ⟨ Include files 13 ⟩
  ⟨ ztstview.web 68 ⟩
  ⟨ Preprocessor macro definitions 11 ⟩
  IMPLEMENT_DYNCREATE(CZTestView, CView)
  BEGIN_MESSAGE_MAP(CZTestView, CView)
  ON_COMMAND(ID_FILE_PRINT, CView :: OnFilePrint)
  ON_COMMAND(ID_FILE_PRINT_DIRECT, CView :: OnFilePrint)
  ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView :: OnFilePrintPreview)
  END_MESSAGE_MAP( )
  ⟨ Define CZTestView functions 75 ⟩
```

96. This is what's written to `ztstview.h`.

```
⟨ ztstview.h 96 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ Declare class CZTestView 72 ⟩
```

97. ZTest (`ztest.web`). [LDF 2006.09.18.]

```
⟨ ztest.web 97 ⟩ ≡
  static char id_string[] = "$Id:␣ztest.web,v␣1.7␣2006/12/11␣10:22:42␣Administrator␣Exp␣$";
```

This code is cited in sections 6 and 8.

This code is used in section 132.

98. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
#if 0    /* 1 */
#include "ZTest.h"
#endif
#include "MainFrm.h"
#include "ZTestDoc.h"
#include "ztstview.h"
#include "DIALOGZ1.H"
```

99. Preprocessor macro definitions.

```
⟨ Preprocessor macro definitions 11 ⟩ +≡
#ifdef _DEBUG
#define new DEBUG_NEW
#endif
```

100. CZTestApp declaration. [LDF Undated.]

```
⟨ CZTestApp declaration 100 ⟩ ≡
  class CZTestApp : public CWinApp {
  public: CZTestApp();
  public: virtual BOOL InitInstance();
    afx_msg void OnAppAbout();
    DECLARE_MESSAGE_MAP()
  };
```

This code is used in section 133.

101. Extern variable declarations. [LDF Undated.]

```
⟨ extern variable declarations 101 ⟩ ≡
  extern CZTestApp theApp;
```

This code is used in section 133.

102. CZTestApp message map. These are macro calls. [LDF 2006.09.14.]

```
⟨ CZTestApp code 102 ⟩ ≡
  BEGIN_MESSAGE_MAP(CZTestApp, CWinApp)
  ON_COMMAND(ID_APP_ABOUT, OnAppAbout)    /* Dateibasierte Standarddokumentbefehle */
  ON_COMMAND(ID_FILE_NEW, CWinApp :: OnFileNew)
  ON_COMMAND(ID_FILE_OPEN, CWinApp :: OnFileOpen)    /* Standarddruckbefehl "Seite einrichten" */
  ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp :: OnFilePrintSetup)
  END_MESSAGE_MAP()
```

See also sections 104, 105, 107, 108, 109, 110, 111, 112, 113, 114, 115, and 116.
This code is used in section 132.

103. CZTestApp Functions. [LDF Undated.]

104. Constructor. [LDF Undated.]

```
⟨ CZTestApp code 102 ⟩ +≡
  CZTestApp :: CZTestApp( )
  {
    return;
  }
```

105. The only **CZTestApp** object. [LDF Undated.]

```
⟨ CZTestApp code 102 ⟩ +≡
  CZTestApp theApp;
```

106. Initialize instance.

107.

Log

[LDF 2006.12.11.] Now calling the global function *init_copyright_strings*.

```
⟨ CZTestApp code 102 ⟩ +≡
  BOOL CZTestApp :: InitInstance( ){
```

108. *InitCommonControls* ist für Windows XP erforderlich, wenn ein Anwendungsmanifest die Verwendung von `ComCtl32.dll` Version 6 oder höher zum Aktivieren von visuellen Stilen angibt. Ansonsten treten beim Erstellen von Fenstern Fehler auf.

```
⟨ CZTestApp code 102 ⟩ +≡
  InitCommonControls( );
  CWinApp :: InitInstance( );
```

109. Initialize OLE libraries.

```
⟨ CZTestApp code 102 ⟩ +≡
  if (¬AfxOleInit( )) {
    AfxMessageBox(IDP_OLE_INIT_FAILED);
    return FALSE;
  }
  AfxEnableControlContainer( );
```

110. Standardinitialisierung. Wenn Sie diese Features nicht verwenden und die Größe der ausführbaren Datei verringern möchten, entfernen Sie die nicht erforderlichen Initialisierungsroutinen. Ändern Sie den Registrierungsschlüssel unter dem Ihre Einstellungen gespeichert sind. TODO: Ändern Sie diese Zeichenfolge entsprechend, z.B. zum Namen Ihrer Firma oder Organisation.

```
⟨ CZTestApp code 102 ⟩ +≡
  SetRegistryKey(_T("Vom␣lokalen␣Anwendungs-Assistenten␣generierte␣Anwendungen"));
  LoadStdProfileSettings(4);      /* Standard INI-Dateioptionen laden (einschließlich MRU) */
```

111. Dokumentvorlagen der Anwendung registrieren. Dokumentvorlagen dienen als Verbindung zwischen Dokumenten, Rahmenfenstern und Ansichten.

```
⟨ CZTestApp code 102 ⟩ +≡
  CSingleDocTemplate *pDocTemplate;
  pDocTemplate = new CSingleDocTemplate(IDR_MAINFRAME, RUNTIME_CLASS(CZTestDoc),
      RUNTIME_CLASS(CMainFrame),      /* Haupt-SDI-Rahmenfenster */
  RUNTIME_CLASS(CZTestView));
  if (¬pDocTemplate) return FALSE;
  AddDocTemplate(pDocTemplate);
```

112. Befehlszeile parsen, um zu prüfen auf Standardumgebungsbefehle DDE, Datei offen

```
⟨ CZTestApp code 102 ⟩ +≡
  CCommandLineInfo cmdInfo;
  ParseCommandLine(cmdInfo);
```

113. Verteilung der in der Befehlszeile angegebenen Befehle. Es wird FALSE zurückgegeben, wenn die Anwendung mit /RegServer, /Register, /Unregserver oder /Unregister gestartet wurde.

```
⟨ CZTestApp code 102 ⟩ +≡
  if (¬ProcessShellCommand(cmdInfo)) return FALSE;
```

114. Das einzige Fenster ist initialisiert und kann jetzt angezeigt und aktualisiert werden.

```
⟨ CZTestApp code 102 ⟩ +≡
  m_pMainWnd→ShowWindow(SW_SHOW);
  m_pMainWnd→UpdateWindow( );
```

115. Rufen Sie DragAcceptFiles nur auf, wenn eine Suffix vorhanden ist. In einer SDI-Anwendung ist dies nach ProcessShellCommand erforderlich

```
⟨ CZTestApp code 102 ⟩ +≡
  init_copyright_strings( );
  Dialog_Z_1 dlg_1;
  dlg_1.DoModal( );
```

116. Now calling *zoomtst2* in **Dialog_Z_1** :: *OnOK*. [LDF 2006.07.21.]

```
⟨ CZTestApp code 102 ⟩ +≡
#if 0      /* This works. [LDF 2006.07.12] . */
  zoomtst0( );
  exit(0);
#endif
  return TRUE; }      /* End of CZTestApp::InitInstance definition. */
```

117. Class CAboutDlg declaration.
CAboutDlg-Dialogfeld für Anwendungsbefehl “Info”. [LDF Undated.]

```
⟨ Declare class CAboutDlg 117 ⟩ ≡
  class CAboutDlg : public CDialog {
public: CAboutDlg( );
```

See also sections 118, 119, and 120.

This code is used in section 132.

118. Dialogfelddaten.

```
⟨ Declare class CAboutDlg 117 ⟩ +≡
  enum {
    IDD = IDD_ABOUTBOX
  };
protected: virtual void DoDataExchange(CDataExchange *pDX);
    /* DDX/DDV-Unterstützung */
```

119. Implementierung.

```
⟨ Declare class CAboutDlg 117 ⟩ +≡
```

120.

```
⟨ Declare class CAboutDlg 117 ⟩ +≡
protected: DECLARE_MESSAGE_MAP( ) } ;
```

121. CAboutDlg Functions. [LDF Undated.]

122. Constructor. [LDF Undated.]

```
⟨ Define CAboutDlg functions 122 ⟩ ≡
  CAboutDlg::CAboutDlg( )
  : CDialog(CAboutDlg::IDD) { }
```

See also sections 123 and 124.

This code is used in section 132.

123. Do Data Exchange. [LDF Undated.]

```
⟨ Define CAboutDlg functions 122 ⟩ +≡
  void CAboutDlg::DoDataExchange(CDataExchange *pDX)
  {
    CDialog::DoDataExchange(pDX);
  }
```

124. CAboutDlg Message map. [LDF Undated.]

```
⟨ Define CAboutDlg functions 122 ⟩ +≡
  BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
  END_MESSAGE_MAP( )
```

125. Anwendungsbefehl zum Ausführen des Dialogfelds [LDF Undated.]

```
⟨ Define CZTestApp functions 125 ⟩ ≡
  void CZTestApp::OnAppAbout( )
  {
    CAboutDlg aboutDlg;
    aboutDlg.DoModal( );
  }
```

This code is used in section 132.

126. CZTestApp Meldungshandler. [LDF Undated.]

127. Global variables. [LDF Undated.]

Log

[LDF 2006.08.23.] Added this section.

[LDF 2006.08.23.] Added the declaration of **CMutex** *time_mutex*.

[LDF 2006.12.11.] Added the declaration of **string** *copyright_html_str*.

⟨ Global variables 18 ⟩ +≡
  CMutex *time_mutex*;
  string *copyright_html_str*;

128. Global functions. [LDF 2006.12.11.]

Log

[2006.12.11.] Added this section.

---

129. Initialize copyright string. [LDF 2006.12.11.]

Log

[2006.12.11.] Added this function.

⟨ Declare global functions 129 ⟩ ≡
  int *init_copyright_strings*(**void**);

This code is used in section 133.

130.

```
⟨ Define global functions 130 ⟩ ≡
  int init_copyright_strings(void)
  {
#if 0     /* 1 */
#define DEBUG_OUTPUT 1
#else
#undef DEBUG_OUTPUT
#endif
    stringstream temp_strm;
#ifdef DEBUG_OUTPUT
    temp_strm.str("Entering␣'init_copyright_strings'.");
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    temp_strm ≪ "<h2␣align=\"center\">\n<a␣name=\"Copyright\">␣Copyright␣and␣License." ≪
        "\n</a>\n</h2>\n" ≪ "<p>\nThis␣file␣was␣generated␣by␣the␣IWF\
        ␣Metadata␣Harvester,␣" ≪ "a␣package␣for␣metadata␣harvesting.␣␣" ≪
        "The␣following␣copyright␣notice␣applies␣to␣this␣file.␣␣" ≪
        "The␣records␣contained␣in␣this␣file␣and␣the␣resources␣to␣which␣they␣refer␣" ≪
        "are␣subject␣to␣their␣own␣copyrights␣and␣licenses.\n</p>\n" ≪
        "<p>\nCopyright␣(C)␣2006,␣2007␣IWF␣Wissen␣und␣Medien␣gGmbH\n</p>\n" ≪
        "<p>\nThe␣author␣is␣Laurence␣D.␣Finston.\n</p>\n" ≪ "<p>\n" ≪ "The␣IWF␣Metadata␣Ha\
        rvester␣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.␣" ≪ "</p>\n" ≪ "<p>\n" ≪
        "The␣IWF␣Metadata␣Harvester␣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.␣" ≪ "</p>\n" ≪ "<p>\n" ≪
        "You␣should␣have␣received␣a␣copy␣of␣the␣GNU␣General␣Public␣License␣" ≪
        "along␣with␣the␣IWF␣Metadata␣Harvester;␣if␣not,␣write␣to␣the␣Free␣Software␣" ≪
        "Foundation,␣Inc.,␣51␣Franklin␣St,␣Fifth␣Floor,␣Boston,␣MA␣␣02110-1301␣␣USA␣" ≪
        "</p>\n" ≪ "<p>\n" ≪ "The␣IWF␣Metadata␣Harvester␣is␣available␣for␣downloading␣fro\
        m␣the␣" ≪ "following␣FTP␣server:␣" ≪ "ftp://ftp.gwdg.de/pub/gnu2/iwfmdh/␣" ≪
        "</p>\n" ≪ "<p>\n" ≪ "Please␣send␣bug␣reports␣to␣lfinsto1@gwdg.de␣" ≪
        "</p>\n" ≪ "<p>\n" ≪ "The␣author␣can␣be␣contacted␣at:␣" ≪ "</p>\n" ≪
        "<p>\n" ≪ "Laurence␣D.␣Finston<br>\n" ≪ "Kreuzbergring␣41<br>\n" ≪
        "D-37075␣Goettingen<br>\n" ≪ "Germany\n" ≪ "</p>\n" ≪ "<p>\n" ≪
        "lfinsto1@gwdg.de<br>\n" ≪ "s246794@stud.uni-goettingen.de\n" ≪ "</p>\n";
    copyright_html_str = temp_strm.str();
#ifdef DEBUG_OUTPUT
    temp_strm.str("Exiting␣'init_copyright_strings'.");
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    return 0;
#undef DEBUG_OUTPUT
  }   /* End of init_copyright_strings definition. */
```

This code is used in section 132.

131. Putting ZTest together. [LDF Undated.]

132. This is what's compiled.

```
⟨ Include files 13 ⟩
⟨ ztest.web 97 ⟩
⟨ Preprocessor macro definitions 11 ⟩
⟨ Global variables 18 ⟩
⟨ CZTestApp code 102 ⟩
⟨ Declare class CAboutDlg 117 ⟩
⟨ Define CAboutDlg functions 122 ⟩
⟨ Define CZTestApp functions 125 ⟩
⟨ Define global functions 130 ⟩
```

133. This is what gets written to `ztest.h`.

```
⟨ ztest.h 133 ⟩ ≡
#pragma once
#ifndef __AFXWIN_H__
  #
  error 'stdafx.h'␣muss␣vor␣dieser␣Datei␣in␣PCH␣eingeschlossen␣werden.
#endif
#include "resource.h"     /* Hauptsymbole */
    ⟨ CZTestApp declaration 100 ⟩
  ⟨ extern variable declarations 101 ⟩
  ⟨ Declare global functions 129 ⟩
```

134. Dialog_ Z_1 (dialogz1.web). [LDF Undated.]

```
⟨ dialogz1.web 134 ⟩ ≡
  static char id_string[] = "$Id:␣dialogz1.web,v␣1.7␣2006/10/23␣13:10:27␣Administrator␣Exp␣$";
```

This code is cited in sections 6 and 8.

This code is used in section 175.

135. Include files. [LDF Undated.]

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
#if 0     /* 1 */
  # include "ZTest.h"
#endif
#include ".\DIALOGZ1.H"
```

136. Dialog_Z_1 class declaration. [LDF Undated.]

```
⟨ Declare class Dialog_Z_1 136 ⟩ ≡
  class Dialog_Z_1 : public CDialog
  {
  DECLARE_DYNAMIC(Dialog_Z_1)
```

See also sections 137 and 138.

This code is used in section 176.

137. Dialogfelddaten. [LDF Undated.]

```
⟨ Declare class Dialog_Z_1 136 ⟩ +≡
public: enum {
    IDD = IDD_DIALOG1
  };
```

138.

```
⟨ Declare class Dialog_Z_1 136 ⟩ +≡
  protected: DECLARE_MESSAGE_MAP( )
        ⟨ Declare protected Dialog_Z_1 functions 145 ⟩
    public: ⟨ Declare public Dialog_Z_1 functions 141 ⟩
        CString search_command;
        BOOL perform_search;
        BOOL parse_records;
        BOOL display_records;
        BOOL write_records_file;
        BOOL clear_database;
        int display_to_record;
        int display_from_record;
        int display_all_or_range;
        static const int ALL;
        static const int RANGE; } ;
```

139. Local static constants for Dialog_Z_1. [LDF Undated.]

```
⟨ Local static constants for Dialog_Z_1 139 ⟩ ≡
  const int Dialog_Z_1 :: ALL = 1;
  const int Dialog_Z_1 :: RANGE = 2;
```

This code is used in section 175.

140. Dialog_Z_1 functions.

141. Constructor. [LDF Undated.]

```
⟨ Declare public Dialog_Z_1 functions 141 ⟩ ≡
  Dialog_Z_1(CWnd *pParent = NULL);
```

See also sections 143, 149, 160, 162, 164, 166, 168, 170, and 172.

This code is used in section 138.

142.

⟨ Define **Dialog_Z_1** functions 142 ⟩ ≡
```
IMPLEMENT_DYNAMIC(Dialog_Z_1, CDialog)
Dialog_Z_1 :: Dialog_Z_1(CWnd *pParent     /* = NULL */ )
  : CDialog(Dialog_Z_1 :: IDD, pParent), perform_search(FALSE), parse_records(FALSE),
        display_records(FALSE), write_records_file(FALSE), clear_database(FALSE), display_to_record(0),
        display_from_record(0), display_all_or_range(0) {
    return;
}
```
See also sections 144, 146, 150, 151, 153, 154, 155, 156, 157, 159, 161, 163, 165, 167, 169, 171, and 173.

This code is used in section 175.

143. Destructor. [LDF Undated.]

⟨ Declare **public Dialog_Z_1** functions 141 ⟩ +≡
```
virtual ~Dialog_Z_1( );
```

144.

⟨ Define **Dialog_Z_1** functions 142 ⟩ +≡
```
Dialog_Z_1 :: ~Dialog_Z_1( )
{ }
```

145. DoDataExchange. [LDF Undated.]

⟨ Declare **protected Dialog_Z_1** functions 145 ⟩ ≡
```
virtual void DoDataExchange(CDataExchange *pDX);
```
See also sections 152 and 158.

This code is used in section 138.

146.

⟨ Define **Dialog_Z_1** functions 142 ⟩ +≡

```
void Dialog_Z_1 :: DoDataExchange(CDataExchange *pDX)
{
  CDialog :: DoDataExchange(pDX);
  DDX_Text(pDX, IDC_SEARCH_COMMAND, search_command);
  DDX_Check(pDX, IDC_PERFORM_SEARCH, perform_search);
  DDX_Check(pDX, IDC_PARSE_RECORDS, parse_records);
  DDX_Check(pDX, IDC_DISPLAY_RECORDS, display_records);
  DDX_Check(pDX, IDC_WRITE_RECORDS_FILE, write_records_file);
  DDX_Check(pDX, IDC_CLEAR_DATABASE, clear_database);
  DDX_LBIndex(pDX, IDC_DISPLAY_TO_RECORD, display_to_record);
  DDX_LBIndex(pDX, IDC_DISPLAY_FROM_RECORD, display_from_record);
}
```

147. Message map. These are macro calls. [LDF Undated.]

⟨ Macro calls for **Dialog_Z_1** message map 147 ⟩ ≡

```
BEGIN_MESSAGE_MAP(Dialog_Z_1, CDialog)
ON_BN_CLICKED(IDC_PERFORM_SEARCH, OnBnClickedPerformSearch)
ON_BN_CLICKED(IDC_PARSE_RECORDS, OnBnClickedParseRecords)
ON_BN_CLICKED(IDC_DISPLAY_RECORDS, OnBnClickedDisplayRecords)
ON_BN_CLICKED(IDC_WRITE_RECORDS_FILE, OnBnClickedWriteRecordsFile)
ON_BN_CLICKED(IDC_CLEAR_DATABASE, OnBnClickedClearDatabase)
ON_BN_CLICKED(IDC_DISPLAY_ALL, OnBnClickedDisplayAll)
ON_BN_CLICKED(IDC_DISPLAY_RANGE, OnBnClickedDisplayRange)
END_MESSAGE_MAP()
```

This code is used in section 175.

148. Event handlers. [LDF Undated.]

149. OnInitDialog. [LDF Undated.]

⟨ Declare **public Dialog_Z_1** functions 141 ⟩ +≡

```
virtual BOOL OnInitDialog();
```

150.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  BOOL Dialog_Z_1 :: OnInitDialog(void){ CListBox *pLB = ( CListBox * ) GetDlgItem(IDC_SERVERS);
      pLB→InsertString(−1, "GBV/GVK");
      pLB→InsertString(−1, "XXX");
      pLB→InsertString(−1, "YYY");
      pLB→InsertString(−1, "ZZZ");
      pLB→SetSel(0, TRUE);
      pLB = NULL;
      CButton *pBT;
      pBT = (CButton *) GetDlgItem(IDC_PERFORM_SEARCH);
#if 1    /* 0 */
      pBT→SetCheck(BST_CHECKED);
      perform_search = TRUE;
#else
      pBT→SetCheck(BST_UNCHECKED);
      perform_search = FALSE;
#endif
      pBT = (CButton *) GetDlgItem(IDC_WRITE_RECORDS_FILE);
#if 0    /* 1 */
      pBT→SetCheck(BST_CHECKED);
      write_records_file = TRUE;
#else
      pBT→SetCheck(BST_UNCHECKED);
      write_records_file = FALSE;
#endif
      pBT = (CButton *) GetDlgItem(IDC_CLEAR_DATABASE);
#if 0    /* 1 */
      pBT→SetCheck(BST_CHECKED);
      clear_database = TRUE;
#else
      pBT→SetCheck(BST_UNCHECKED);
      clear_database = FALSE;
#endif
      pBT = (CButton *) GetDlgItem(IDC_PARSE_RECORDS);
#if 0    /* 1 */
      pBT→SetCheck(BST_CHECKED);
      parse_records = TRUE;
#else
      pBT→SetCheck(BST_UNCHECKED);
      parse_records = FALSE;
#endif
      pBT = (CButton *) GetDlgItem(IDC_DISPLAY_RECORDS);
#if 0    /* 1 */
      pBT→SetCheck(BST_CHECKED);
      display_records = TRUE;
#else
      pBT→SetCheck(BST_UNCHECKED);
      display_records = FALSE;
#endif
      pBT = (CButton *) GetDlgItem(IDC_DISPLAY_ALL);
```

```
        pBT→SetCheck(BST_CHECKED);
        display_all_or_range = ALL;
        pBT = NULL;
        CEdit *pEB = (CEdit *) GetDlgItem(IDC_SEARCH_COMMAND);
        pEB→SetFocus( );
#if 0
        search_command = "@and␣@attr␣1=1␣cundy␣@attr␣1=31␣1981";
        search_command = "@and␣@and␣@attr␣1=4␣Elektronische␣@attr␣1=4␣";
        search_command += "Ressource␣@attr␣1=1031␣dvd-video";
        search_command = "@and␣@and␣@attr␣1=59␣Clausthal␣@attr␣1=1031␣@attr␣4=1␣";
        search_command += "\"Elektronische␣Ressource\"␣@attr␣1=31␣@attr␣2=4␣@attr␣4=4␣2006";
        search_command = "@and␣@and␣@attr␣1=59␣Clausthal␣@attr␣1=1031␣@attr␣4=1␣";
        search_command += "\"Elektronische␣Ressource\"␣@attr␣1=101\
            1␣@attr␣2=3␣"search_command += "@attr␣4=5␣20060801";
        search_command = "@and␣@and␣@attr␣1=59␣Clausthal␣@attr␣1=1031␣";
#endif
        search_command = "@and␣@and␣@attr␣1=59␣Clausthal␣@attr␣1=1031␣";
        search_command += "@attr␣4=1␣\"Elektronische␣Ressource\"␣@attr␣1=31␣";
        search_command += "@attr␣2=3␣@attr␣4=4␣1999";
        UpdateData(FALSE);
#if 0    /* 1 */
        pEB→UpdateWindow( );
#endif
```

151. return TRUE unless you set the focus to a control. AUSNAHME: OCX-Eigenschaftenseite muss FALSE zurückgeben.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
#if 0    /* 1 */
  return TRUE;
#endif
  return CDialog :: OnInitDialog( ); }    /* End of Dialog_Z_1 :: OnInitDialog( ) definition. */
```

152. On OK.

Log

[LDF Undated.] Added this function.

```
⟨ Declare protected Dialog_Z_1 functions 145 ⟩ +≡
  virtual void OnOK( );
```

153.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  void Dialog_Z_1 :: OnOK ( ){ stringstream temp_strm;
      CEdit *pEB = (CEdit *) GetDlgItem(IDC_SEARCH_COMMAND); CListBox * pLB = ( CListBox * )
          GetDlgItem(IDC_SERVERS);
      int buff[16];
      int items_ctr;
      items_ctr = pLB→GetSelItems(16, buff);
      temp_strm ≪ "‘items_ctr’␣==␣" ≪ items_ctr;
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
      int select_value;
      unsigned short server_selector = 0;
      if (items_ctr ≡ LB_ERR) {
        temp_strm ≪ "ERROR!␣␣In␣‘Dialog_Z_1::OnOK’:" ≪ endl ≪
            "Invalid␣search␣field,␣‘items_ctr’␣==␣" ≪ items_ctr ≪ endl ≪
            "Exiting␣function␣unsuccessfully␣(no␣return␣value).";
        AfxMessageBox(temp_strm.str( ).c_str( ));
        temp_strm.str("");
        return;
      }
```

154.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  else
    if (items_ctr ≡ 0) {
      temp_strm ≪ "WARNING!␣␣In␣‘Dialog_Z_1::OnOK’:" ≪ endl ≪
          "No␣servers␣field␣selected,␣‘items_ctr’␣==␣" ≪ items_ctr ≪ endl ≪
          "Please␣pick␣one␣or␣more␣servers.";
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
    }
```

155.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  else    /* ¬(items_ctr ≡ LB_ERR ∨ items_ctr ≡ 0) */
  {
    select_value = 0;
    CString selection_str;
    for (int i = 0; i < items_ctr; ++i) {
      pLB→GetText(buff[i], selection_str);
#if 1
      temp_strm ≪ "buff[" ≪ i ≪ "]␣==␣" ≪ buff[i] ≪ endl ≪ "selection_str␣==␣" ≪
          selection_str ≪ endl;
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
#endif
      if (selection_str ≡ "GBV/GVK") {
#if 1     /* 0 */
        temp_strm ≪ "Connecting␣with␣the␣server␣GBV/GVK.";
        AfxMessageBox(temp_strm.str( ).c_str( ));
        temp_strm.str("");
#endif
        server_selector |= ZClient::GBV_GVK_ID;
      }
      else if (selection_str ≡ "XXX") {
#if 1     /* 0 */
        temp_strm ≪ "Connecting␣with␣the␣server␣XXX.";
        AfxMessageBox(temp_strm.str( ).c_str( ));
        temp_strm.str("");
#endif
      }
      else {
#if 1     /* 0 */
        temp_strm ≪ "Invalid␣server␣name.";
        AfxMessageBox(temp_strm.str( ).c_str( ));
        temp_strm.str("");
#endif
      }
    }    /* for */
  }    /* else (¬(items_ctr ≡ LB_ERR ∨ items_ctr ≡ 0)) */
  while (true) {
    pEB→GetWindowText(search_command);
    if (search_command ≡ "") {
      AfxMessageBox("Please␣enter␣a␣search␣command.");
      return;
    }
    else break;
  }    /* while */
#if 0     /* 1 */
  temp_strm ≪ "search_command␣==␣" ≪ search_command;
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
```

```
#if 1    /* 0 */
```

156.

Log

[LDF 2006.09.25.] No longer casting the arguments to *zoomtst2* to **bool**, since I changed them to **BOOL**. Using **bool** caused Microsoft Visual Studio to issue warnings.

[LDF 2006.10.16.] Now passing **unsigned short** *server_selector* to *zoomtst2*.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  zoomtst2(server_selector, search_command, perform_search, write_records_file, clear_database,
      parse_records, display_records);
#endif
```

157.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
#if 0    /* 1 */
  search_command = "@and␣@and␣@attr␣1=4␣Elektronische␣@attr␣1=4␣Ressource␣";
  search_command += "@attr␣1=1031␣dvd-video";
#endif
  UpdateData(FALSE);
#if 0    /* 1 */
  pEB→UpdateWindow();
  pEB→SetSel(0, −1);
  pEB→Clear();
  pEB→SetFocus();
  UpdateData(FALSE);
  CDialog :: OnOK();
#endif
  }    /* End of Dialog_Z_1 :: OnOK definition. */
```

158. OnCancel. [LDF Undated.]

```
⟨ Declare protected Dialog_Z_1 functions 145 ⟩ +≡
  virtual void OnCancel();
```

159.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  void Dialog_Z_1 :: OnCancel()
  {
#if 0    /* 1 */
    CDialog :: OnCancel();
#endif
    exit(0);
  }
```

160. OnBnClickedPerformSearch.

Log

[LDF 2006.08.31.] Added this function.

```
⟨ Declare public Dialog_Z_1 functions 141 ⟩ +≡
  afx_msg void OnBnClickedPerformSearch( );
```

161.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  void Dialog_Z_1 :: OnBnClickedPerformSearch( )
  {
    stringstream temp_strm;
    perform_search = ¬perform_search;
#if 0    /* 1 */
    temp_strm ≪ "‘perform_search’␣==␣" ≪ perform_search;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    return;
  }    /* End of Dialog_Z_1 :: OnBnClickedPerformSearch definition. */
```

162. OnBnClickedParseRecords. [LDF 2006.08.31.]

Log

[LDF 2006.08.31.] Added this function.

```
⟨ Declare public Dialog_Z_1 functions 141 ⟩ +≡
  afx_msg void OnBnClickedParseRecords( );
```

163.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  void Dialog_Z_1 :: OnBnClickedParseRecords(void)
  {
    stringstream temp_strm;
    parse_records = ¬parse_records;
#if 0    /* 1 */
    temp_strm ≪ "‘parse_records’␣==␣" ≪ parse_records;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    return;
  }    /* End of Dialog_Z_1 :: OnBnClickedParseRecords definition. */
```

164. OnBnClickedDisplayRecords. [LDF 2006.08.31.]

Log

[LDF 2006.08.31.] Added this function.

```
⟨ Declare public Dialog_Z_1 functions 141 ⟩ +≡
  afx_msg void OnBnClickedDisplayRecords( );
```

165.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  void Dialog_Z_1 :: OnBnClickedDisplayRecords( )
  {
    stringstream temp_strm;
    display_records = ¬display_records;
#if 0      /* 1 */
    temp_strm ≪ "‘display_records’␣==␣" ≪ display_records;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    return;
  }      /* End of Dialog_Z_1 :: OnBnClickedDisplayRecords definition. */
```

166. OnBnClickedWriteRecordsFile. [LDF 2006.08.31.]

Log

[LDF 2006.08.31.] Added this function.

```
⟨ Declare public Dialog_Z_1 functions 141 ⟩ +≡
  afx_msg void OnBnClickedWriteRecordsFile( );
```

167.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  void Dialog_Z_1 :: OnBnClickedWriteRecordsFile( )
  {
    stringstream temp_strm;
    write_records_file = ¬write_records_file;
#if 0      /* 1 */
    temp_strm ≪ "‘write_records_file’␣==␣" ≪ write_records_file;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    return;
  }      /* End of Dialog_Z_1 :: OnBnClickedWriteRecordsFile definition. */
```

168. OnBnClickedClearDatabase. [LDF 2006.08.31.]

Log

[LDF 2006.08.31.] Added this function.

```
⟨ Declare public Dialog_Z_1 functions 141 ⟩ +≡
  afx_msg void OnBnClickedClearDatabase( );
```

169.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  void Dialog_Z_1 :: OnBnClickedClearDatabase( )
  {
    stringstream temp_strm;
    clear_database = ¬clear_database;
#if 0    /* 1 */
    temp_strm ≪ "‘clear_database’␣==␣" ≪ clear_database;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    return;
  }    /* End of Dialog_Z_1 :: OnBnClickedClearDatabase definition. */
```

170. OnBnClickedDisplayAll.

Log

[LDF 2006.09.04.] Added this function.

```
⟨ Declare public Dialog_Z_1 functions 141 ⟩ +≡
  afx_msg void OnBnClickedDisplayAll( );
```

171.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  void Dialog_Z_1 :: OnBnClickedDisplayAll(void)
  {
    display_all_or_range = ALL;
  }
```

172. OnBnClickedDisplayRange. [LDF 2006.09.04.]

Log

[LDF 2006.09.04.] Added this function.

```
⟨ Declare public Dialog_Z_1 functions 141 ⟩ +≡
  afx_msg void OnBnClickedDisplayRange( );
```

173.

```
⟨ Define Dialog_Z_1 functions 142 ⟩ +≡
  void Dialog_Z_1 :: OnBnClickedDisplayRange(void)
  {
    display_all_or_range = RANGE;
  }
```

174. Putting Dialog_Z_1 together.

175. This is what gets compiled.

```
  ⟨ Include files 13 ⟩
  ⟨ dialogz1.web 134 ⟩
  ⟨ Local static constants for Dialog_Z_1 139 ⟩
  ⟨ Define Dialog_Z_1 functions 142 ⟩
  ⟨ Macro calls for Dialog_Z_1 message map 147 ⟩
```

176. This is what gets written to `dialogz1.h`. [LDF Undated.]

```
⟨ dialogz1.h 176 ⟩ ≡
#pragma once
#include "resource.h"
  ⟨ Declare class Dialog_Z_1 136 ⟩
```

177. Scanning. (`scanner.web`). [LDF 2006.10.17.]

Log

[LDF 2006.10.17.] Created this file.

```
⟨ scanner.web 177 ⟩ ≡
  static char id_string[] = "$Id:␣scanner.web,v␣1.6␣2006/10/23␣12:44:35␣Administrator␣Exp␣$";
```

This code is cited in sections 6 and 8.

This code is used in section 186.

178. Include files.

```
⟨ Include files 13 ⟩ +≡
#if 0     /* 1 */
#include "stdafx.h"
#endif
#include <stdlib.h>
#include <stdio.h>
```

179. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#if 0     /* 1 */
#pragma once
#endif
```

180. Type declarations for the scanner. [LDF 2006.10.17.]

181. Union declaration for YYSTYPE. [LDF 2006.10.17.]

```
⟨ Union declaration for YYSTYPE 181 ⟩ ≡
  struct MDH_YYSTYPE {
    union {
      char string_value[64];
      signed int int_value;
      void *pointer_value;
    } u;
  };
  typedef struct MDH_YYSTYPE YYSTYPE;
```

This code is used in sections 186 and 187.

182. Location type. [LDF 2006.10.17.]

```
⟨ Declare location type 182 ⟩ ≡
  struct MDH_YYLTYPE {
    unsigned int first_line;
    unsigned int first_column;
    unsigned int last_line;
    unsigned int last_column;
    unsigned int position;
  };
  typedef struct MDH_YYLTYPE YYLTYPE;
```

This code is used in sections 186 and 187.

183. yylex. [LDF 2006.10.17.]

```
⟨ Declare yylex 183 ⟩ ≡
  int yylex(YYSTYPE *value, YYLTYPE *location, void *parameter);
```

This code is used in section 187.

184.

```
⟨ Define yylex 184 ⟩ ≡
  int yylex(YYSTYPE *value, YYLTYPE *location, void *parameter)
  {
    printf("In␣‘yylex’.\n");
    return 0;
  }    /* End of yylex definition. */
```

This code is used in section 186.

185. Putting the scanner together.

186. This is what's compiled.

```
  ⟨ Include files 13 ⟩
  ⟨ scanner.web 177 ⟩
  ⟨ Union declaration for YYSTYPE 181 ⟩
  ⟨ Declare location type 182 ⟩
  ⟨ Define yylex 184 ⟩
```

187. This is what's written to `scanner.h`.

⟨ `scanner.h` 187 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ Union declaration for **YYSTYPE** 181 ⟩
  ⟨ Declare location type 182 ⟩
  ⟨ Declare *yylex* 183 ⟩

188. ZClient (`zclient.web`). [LDF 2006.09.18.]

⟨ `zclient.web` 188 ⟩ ≡
  static char *id_string*[ ] = `"$Id:␣zclient.web,v␣1.8␣2006/11/28␣16:25:12␣Administrator␣Exp␣$"`;

This code is cited in sections 6 and 8.

This code is used in section 362.

189. Include files.

⟨ Include files 13 ⟩ +≡
#include `"stdafx.h"`
#include `"MainFrm.h"`
#include `"ZTestDoc.h"`
#include `"ztstview.h"`

190. Preprocessor macro calls. [LDF Undated.]

⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma *once*

191. using declarations for namespaces. [LDF 2006.09.18.]

⟨ **using** declarations for namespaces 191 ⟩ ≡
  using namespace std;

See also sections 366, 398, 766, 780, 1000, 1021, 1042, 1063, 1084, 1105, 1126, 1147, 1168, 1189, 1210, 1231, 1252, 1273, 1294, 1315, 1336, 1357, 1378, 1399, 1420, 1441, 1462, 1483, 1504, 1525, 1546, 1567, 1588, 1609, 1630, 1651, 1672, and 1693.

This code is used in sections 363, 382, 383, 416, 776, 995, 1017, 1038, 1059, 1080, 1101, 1122, 1143, 1164, 1185, 1206, 1227, 1248, 1269, 1290, 1311, 1332, 1353, 1374, 1395, 1416, 1437, 1458, 1479, 1500, 1521, 1542, 1563, 1584, 1605, 1626, 1647, 1668, 1689, and 1710.

192. ZClient class declaration. [LDF Undated.]

Log

[LDF 2006.07.26.] Added the declaration of *init_category_map*.

[LDF 2006.09.12.] Removed the declaration of *parse_record*. It wasn't needed, and there was no definition for it.

[LDF 2006.10.16.] Changed the type of `GBV_GVK_ID` from **static const long** to **static const unsigned short**. I don't know why it was **long** in the first place.

[LDF 2006.10.16.] Changed the type of *source_id* from **long** to **unsigned short**.

⟨ Declare **class ZClient** 192 ⟩ ≡
  class ZClient
  {
  friend Pica_Record;
public: ⟨ Declare **ZClient** functions 197 ⟩
  static const unsigned short `MAX_ZOOM_CONNECTION`;

```
  static const unsigned short MAX_ZOOM_RESULTSET;
  static const unsigned short GBV_GVK_ID;
```

See also section 193.

This code is used in section 363.

193.

Log

[LDF 2006.09.06.] Changed name of **ofstream** *log_file* to *log_strm*.

[LDF 2006.09.06.] Changed the type of **ofstream** *log_strm* to **Output_Stream_Type**.

```
⟨ Declare class ZClient 192 ⟩ +≡
protected: vector⟨ZOOM_connection *⟩ connections;
  vector⟨ZOOM_resultset *⟩ resultsets;
  Category_Map_Type category_map;
  Output_Stream_Type log_strm;
  Sources source_recordset;
  CDatabase *database;
  unsigned short source_id; } ;
```

194. Initialize static constants. [LDF 2006.09.18.]

```
⟨ Initialize static constants for ZClient 194 ⟩ ≡
  const unsigned short ZClient::MAX_ZOOM_CONNECTION = 16;
  const unsigned short ZClient::MAX_ZOOM_RESULTSET = 16;
  const unsigned short ZClient::GBV_GVK_ID = 1;
```

This code is used in section 362.

195. Functions. [LDF Undated.]

196. Constructors. [LDF 2006.09.18.]

197. Default Constructor. [LDF 2006.09.18.]
The definition is currently empty. [LDF 2006.07.19.]

```
⟨ Declare ZClient functions 197 ⟩ ≡
  ZClient(void);
```

See also sections 199, 206, 209, 211, 214, 216, 218, 220, 222, 224, 327, 329, and 358.

This code is used in section 192.

198.

```
⟨ Define ZClient functions 198 ⟩ ≡
  ZClient::ZClient(void)
  {
    return;
  }
```

See also sections 200, 201, 202, 203, 204, 205, 207, 210, 212, 215, 217, 219, 221, 223, 328, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 359, and 360.

This code is used in section 362.

199. Non-Default Constructor. [LDF Undated.]

Log

[LDF 2006.09.06.] Changed name of **ofstream** *log_file* data member to *log_strm*.

[LDF 2006.09.06.] Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

[LDF 2006.10.16.] Changed the type of the argument *ssource_id* from **long** to **unsigned short**.

```
⟨ Declare ZClient functions 197 ⟩ +≡
  ZClient(unsigned short ssource_id, char *server_name, int port_num = 0);
```

200.

```
⟨ Define ZClient functions 198 ⟩ +≡
  ZClient::ZClient(unsigned short ssource_id, char *server_name, int port_num){
        source_id = ssource_id;
     log_strm.lock();
     log_strm.output_stream.open("log.txt");
     log_strm.unlock(); int error = 0;
     const char *errmsg = "";
     const char *addinfo = "";
     const char *diagset = "";
```

201. Create connection (don't connect yet). [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  connections.push_back(static_cast⟨ZOOM_connection *⟩(malloc(sizeof(ZOOM_connection))));
  ZOOM_connection *curr_connection = connections.back();
  *curr_connection = ZOOM_connection_create(0);
```

202. option: set sru/get operation (only applicable if http: is used). [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
#if 0   /* 1 */
  ZOOM_connection_option_set(*curr_connection, "sru", "post");
#endif
```

203. option: set async operation. [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  ZOOM_connection_option_set(*curr_connection, "async", "1");
```

204. connect to target and initialize. [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  ZOOM_connection_connect(*curr_connection, server_name, port_num);
```

205.

```
⟨ Define ZClient functions 198 ⟩ +≡      /* START HERE: LDF 2006.07.24. Add error handling. */
  source_recordset.Open( );
  database = source_recordset.m_pDatabase;
  database→SetLoginTimeout(120);
  database→SetQueryTimeout(120);
  return; }     /* End of ZClient non-default constructor definition. */
```

206. Destructor. [LDF Undated.]

Log

[LDF 2006.07.19.] Now closing *log_strm*.

[LDF 2006.09.06.] Changed name of **ofstream** *log_file* data member to *log_strm*.

[LDF 2006.09.06.] Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

```
⟨ Declare ZClient functions 197 ⟩ +≡
  ~ZClient(void);
```

207.

```
⟨ Define ZClient functions 198 ⟩ +≡
  ZClient :: ~ZClient(void)
  {
    log_strm.lock( );
    log_strm.output_stream.close( );
    log_strm.unlock( );
    for (Category_Map_Type :: iterator iter = category_map.begin( ); iter ≠ category_map.end( );
          ++iter) delete iter→second;
    category_map.clear( );
    source_recordset.Close( );
    database→Close( );
  }     /* End of ~ZClient definition. */
```

208. Connections.

Log

[LDF 2006.09.25.] Added this section.

209. Get connection. [LDF Undated.]

```
⟨ Declare ZClient functions 197 ⟩ +≡
  ZOOM_connection *ZClient :: get_connection(vector⟨ZOOM_connection *⟩ :: size_type
      connection_ctr);
```

210.

```
⟨ Define ZClient functions 198 ⟩ +≡
  ZOOM_connection *ZClient::get_connection(vector⟨ZOOM_connection *⟩::size_type
          connection_ctr)
  {
    if (connections.size() > connection_ctr) return connections[connection_ctr];
    else return 0;
  }
```

211. Get last connection. [LDF Undated.]

```
⟨ Declare ZClient functions 197 ⟩ +≡
  ZOOM_connection *get_last_connection(void);
```

212.

```
⟨ Define ZClient functions 198 ⟩ +≡
  ZOOM_connection *ZClient::get_last_connection(void)
  {
    if (connections.size() ≡ 0) return 0;
    else return connections.back();
  }
```

213. Resultsets.

Log

[LDF 2006.09.25.] Added this section.

214. Get resultset. [LDF Undated.]

```
⟨ Declare ZClient functions 197 ⟩ +≡
  ZOOM_resultset *get_resultset(vector⟨ZOOM_resultset *⟩::size_type resultset_ctr);
```

215.

```
⟨ Define ZClient functions 198 ⟩ +≡
  ZOOM_resultset *ZClient::get_resultset(vector⟨ZOOM_resultset *⟩::size_type resultset_ctr)
  {
    if (resultsets.size() < resultset_ctr) return resultsets[resultset_ctr];
    else return 0;
  }
```

216. Get last resultset. [LDF Undated.]

```
⟨ Declare ZClient functions 197 ⟩ +≡
  ZOOM_resultset *get_last_resultset(void);
```

217.

```
⟨ Define ZClient functions 198 ⟩ +≡
  ZOOM_resultset *ZClient::get_last_resultset(void)
  {
    if (resultsets.size() ≡ 0) return 0;
    else return resultsets.back();
  }
```

218. Get resultset size. [LDF Undated.]

To Do

Add exception handling for the case that resultset is invalid, or a note that I should add it. [LDF Undated.]

```
⟨ Declare ZClient functions 197 ⟩ +≡
  size_t get_resultset_size(vector⟨ZOOM_resultset *⟩::size_type resultset_ctr);
```

219.

```
⟨ Define ZClient functions 198 ⟩ +≡
  size_t ZClient::get_resultset_size(vector⟨ZOOM_resultset *⟩::size_type resultset_ctr)
  {
    if (resultsets.size( ) < resultset_ctr) return ZOOM_resultset_size(*(resultsets[resultset_ctr]));
    else return 0;
  }
```

220. Get last resultset size. [LDF Undated.]

To Do

Add exception handling for the case that resultset is invalid, or a note that I should add it. [LDF Undated.]

```
⟨ Declare ZClient functions 197 ⟩ +≡
  size_t get_last_resultset_size(void);
```

221.

```
⟨ Define ZClient functions 198 ⟩ +≡
  size_t ZClient::get_last_resultset_size(void)
  {
    if (resultsets.size( ) ≡ 0) return 0;
    else return ZOOM_resultset_size(*get_last_resultset( ));
  }
```

222. Clear database. [LDF 2006.08.23.]

Log

[LDF 2006.08.23.] Added this function.

```
⟨ Declare ZClient functions 197 ⟩ +≡
  int clear_database(void);
```

223.

```
⟨ Define ZClient functions 198 ⟩ +≡
  int ZClient::clear_database(void)
  {
    stringstream sql_strm;
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣‘ZClient::clear_database’.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    if (database ∧ database→IsOpen()) {
      sql_strm ≪ "exec␣delete_tables␣exec␣regenerate_tables";
#if 0     /* 1 */
      temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      database→ExecuteSQL(sql_strm.str().c_str());
      sql_strm.str("");
      return 0;
    }     /* if (database ∧ database→IsOpen()) */
#if 0     /* 1 */
    else {
      temp_strm ≪ "In␣‘ZClient::clear_database’:␣␣" ≪
          "‘database’␣isn’t␣open.␣␣Can’t␣clear␣it.␣␣Returning␣1.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
      return 1;
    }     /* else */
#endif
    return 0;
  }     /* End of ZClient::clear_database definition. */
```

224. Initialize *category_map*. [LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this function. It contains code that used to be in the non-default **ZClient** constructor.

```
⟨ Declare ZClient functions 197 ⟩ +≡
  int init_category_map(void);
```

225.

```
⟨ Define ZClient :: init_category_map 225 ⟩ ≡
  int ZClient :: init_category_map(void){ Category_Container *curr_category_container = 0;
      Subcategory_Container *curr_subcategory_container = 0;
      Database_Command *curr_database_command = 0;
```

See also sections 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, and 326.

This code is used in section 362.

226. 001@ — Unknown Category.
Unbekannte Kategorie.
Not in “Overview” (“Übersicht”).
[LDF 2006.07.20.]

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "001@";
  curr_category_container→pica_3_category_id = "????";
  curr_category_container→content_description_german = "Unbekannte␣Kategorie";
  curr_category_container→content_description_english = "Unknown␣Category";
```

227. ‘0’ — Unknown Field. (Unbekanntes Feld).

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '0';
  curr_subcategory_container→content_description_german = "Unbekanntes␣Feld";
  curr_subcategory_container→content_description_english = "Unknown␣Field";
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
  category_map["001@"] = curr_category_container;
```

228. 001A — Identifier and Date of the Original Catalogue Entry.
Kennung und Datum der Ersterfassung.
[LDF 2006.07.20.]

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "001A";
  curr_category_container→pica_3_category_id = "0200";
  curr_category_container→content_description_german = "Kennung␣und␣Datum␣der␣Ersterfassung";
  curr_category_container→content_description_english = "Identifier␣and␣Date␣of␣the␣Original␣Cat\
      alogue␣Entry";
```

229. '0' — Identifier and Date of the Original Catalogue Entry.
Kennung und Datum der Ersterfassung.
[LDF 2006.07.20.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '0';
  curr_subcategory_container→content_description_german = "Kennung␣und␣Datum␣der␣Ersterfassung";
  curr_subcategory_container→content_description_english = "Identifier␣and␣Date␣of␣the␣Original␣Cat\
      alogue␣Entry";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_001A_0;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
  category_map["001A"] = curr_category_container;
```

230. 001B — Identifier and Date of the Most Recent Change.
Kennung und Datum der letzten Änderung.
[LDF 2006.07.20.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "001B";
  curr_category_container→pica_3_category_id = "0210";
  curr_category_container→content_description_german = "Kennung␣und␣Datum␣der␣letzten␣Aenderung";
  curr_category_container→content_description_english = "Identifier␣and␣Date␣of␣the␣Most␣Recent␣\
      Change";
```

231. '0' — Identifier and Date of the Most Recent Change.
Kennung und Datum der letzten Änderung.
Identifier and Date of the Change.
Kennung und Datum der Änderung.
[LDF 2006.07.20.]

Log

[LDF 2006.08.03.] Added code for adding a **Database_Command** that references **Subcategory_Container**::*f_001B_0* to *curr_subcategory_container*.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '0';
  curr_subcategory_container→content_description_german =
      "Kennung␣und␣Datum␣der␣letzten␣Aenderung";
  curr_subcategory_container→content_description_english = "Identifier␣and␣Date␣of␣the␣Most␣Recent␣\
      Change";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_001B_0;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
```

232. ‘t’ — Time.
Uhrzeit.
[LDF 2006.07.20.]

Log

[LDF 2006.08.03.] Added code for adding a **Database_Command** that references **Subcategory_Container** ::*f_001B_t* to *curr_subcategory_container*.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 't';
  curr_subcategory_container→content_description_german = "Uhrzeit";
  curr_subcategory_container→content_description_english = "Time";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_001B_t;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['t'] = curr_subcategory_container;
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_001B;
  curr_category_container→database_commands.push_back(curr_database_command);
  category_map["001B"] = curr_category_container;
```

233. 001D — Identifier and Date of the Status Change.
Kennung und Datum der Statusänderung.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "001D";
  curr_category_container→pica_3_category_id = "0230";
  curr_category_container→content_description_german = "Kennung␣und␣Datum␣der␣Statusaenderung";
  curr_category_container→content_description_english = "Identifier␣and␣Date␣of␣the␣Status␣Change";
```

234. ‘0’ — Identifier and Date of the Change.
Kennung und Datum der Änderung.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

[LDF 2006.08.03.] Added code for adding a **Database_Command** that references **Subcategory_Container** ::*f_001D_0* to *curr_subcategory_container*.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '0';
  curr_subcategory_container→content_description_german = "Kennung␣und␣Datum␣der␣Aenderung";
  curr_subcategory_container→content_description_english = "Identifier␣and␣Date␣of␣the␣Change";
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_001D_0;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  category_map["001D"] = curr_category_container;
```

235. 001X — Unknown Category.
Unbekannte Kategorie.
Not in “Overview” (“Übersicht”).
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "001X";
  curr_category_container→pica_3_category_id = "????";
  curr_category_container→content_description_german = "Unbekannte␣Kategorie";
  curr_category_container→content_description_english = "Unknown␣Category";
```

236. ‘0’ — Unknown Field.
Unbekanntes Feld.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '0';
  curr_subcategory_container→content_description_german = "Unbekanntes␣Feld";
  curr_subcategory_container→content_description_english = "Unknown␣Field";
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
  category_map["001X"] = curr_category_container;
```

237. 002@ — Bibliographical Type and Status.
Bibliographische Gattung und Status.
Pica3 0500.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "002@";
  curr_category_container→pica_3_category_id = "0500";
  curr_category_container→content_description_german = "Bibliographische␣Gattung␣und␣Status";
  curr_category_container→content_description_english = "Bibliographic␣Type␣and␣Status";
```

238. ‘0’ — Bibliographical Type and Status.
Bibliographische Gattung und Status.
002@ (Pica+), 0500 (Pica3).
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

[LDF 2006.08.03.] Added code for adding a **Database_Command** that references **Subcategory_Container**::*f_002_AT_0* to *curr_subcategory_container*.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '0';
  curr_subcategory_container→content_description_german = "Bibliographische␣Gattung␣und␣Status";
  curr_subcategory_container→content_description_english = "Bibliographic␣Type␣and␣Status";
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_002_AT_0;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  category_map["002@"] = curr_category_container;
```

239. @ — Identification Number (PPN).
Identifikationsnummer (PPN).
Pica+ 003@ / Pica3 0100.
[LDF 2006.07.24.]
!! No information sheet for this category in the "Categorization Guidelines" ("Katagorisierungsrichtlinien")! [LDF 2006.07.24.]
"PPN" = "Pica Production Number". [LDF 2006.08.23.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "003@";
  curr_category_container→pica_3_category_id = "0100";
  curr_category_container→content_description_german = "Identifikationsnummer␣(PPN)";
  curr_category_container→content_description_english = "Identification␣Number␣(PPN)";
```

240. '0' — Identification Number (PPN).
Identifikationsnummer (PPN)
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '0';
  curr_subcategory_container→content_description_german = "Identifikationsnummer␣(PPN)";
  curr_subcategory_container→content_description_english = "Identification␣Number␣(PPN)";
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_003_AT_0;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  category_map["003@"] = curr_category_container;
```

241. 009P — Remote access to electronic resources.
Information regarding remote access to electronic resources.
Angaben zum Zugriff auf elektronische Ressourcen im Fernzugriff.
Pica+ 009P, Pica3 4083.
[LDF 2006.09.07.]

This is the first place where 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

I call this "reusing" **Category_Container** and **Subcategory_Container** functions.

The fields are the same, and the data is written to the same tables, i.e., **Remote_Access** and **Records_Remote_Access**. [LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "009P";
  curr_category_container→pica_3_category_id = "4083";
  curr_category_container→content_description_german = "Angaben␣zum␣Zugriff␣auf␣elektronische␣R\
      essourcen␣im␣Fernzugriff";
  curr_category_container→content_description_english = "Information␣regarding␣remote␣access␣to␣\
      electronic␣resources";
```

242. '0' — Format. [LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '0';
  curr_subcategory_container→content_description_german = "Format";
  curr_subcategory_container→content_description_english = "Format";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209R_0;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
```

243. 'a' — URL (Universal Resource Locator). [LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
   curr_subcategory_container = new Subcategory_Container;
   curr_subcategory_container→pica_plus_field_id = 'a';
   curr_subcategory_container→content_description_german = "URL␣(Universal␣Resource␣Locator)";
   curr_subcategory_container→content_description_english = "URL␣(Universal␣Resource␣Locator)";
   curr_database_command = new Database_Command;
   curr_database_command→function = Subcategory_Container::f_209R_a;
   curr_subcategory_container→database_commands.push_back(curr_database_command);
   curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

244. 'g' — URN (Universal Resource Name). [LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
   curr_subcategory_container = new Subcategory_Container;
   curr_subcategory_container→pica_plus_field_id = 'g';
   curr_subcategory_container→content_description_german = "URN␣(Universal␣Resource␣Name)";
   curr_subcategory_container→content_description_english = "URN␣(Universal␣Resource␣Name)";
   curr_database_command = new Database_Command;
   curr_database_command→function = Subcategory_Container::f_209R_g;
   curr_subcategory_container→database_commands.push_back(curr_database_command);
   curr_category_container→subcategory_map['g'] = curr_subcategory_container;
```

245. 'S' — License indicator.
Lizenzindikator.
[LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
   curr_subcategory_container = new Subcategory_Container;
   curr_subcategory_container→pica_plus_field_id = 'S';
   curr_subcategory_container→content_description_german = "Lizenzindikator";
   curr_subcategory_container→content_description_english = "License␣indicator";
   curr_database_command = new Database_Command;
   curr_database_command→function = Subcategory_Container::f_209R_S;
   curr_subcategory_container→database_commands.push_back(curr_database_command);
   curr_category_container→subcategory_map['S'] = curr_subcategory_container;
```

246. 'x' — Internal Remarks.
Interne Bemerkungen.
[LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'x';
  curr_subcategory_container→content_description_german = "Interne␣Bemerkungen";
  curr_subcategory_container→content_description_english = "Internal␣Remarks";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209R_x;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['x'] = curr_subcategory_container;
```

247. 'y' — Text for the Web Display.
Text für die Web-Anzeige.
[LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'y';
  curr_subcategory_container→content_description_german = "Text␣fuer␣die␣Web-Anzeige";
  curr_subcategory_container→content_description_english = "Text␣for␣the␣Web␣Display";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209R_y;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['y'] = curr_subcategory_container;
```

248.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_209R;
  curr_category_container→database_commands.push_back(curr_database_command);
  category_map["009P"] = curr_category_container;
```

249. 010@ — Code(s) for Languages.
Code(s) für Sprachen.
Pica+ 010@, Pica3 1500.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "010@";
  curr_category_container→pica_3_category_id = "1500";
  curr_category_container→content_description_german = "Code(s)␣fuer␣Sprachen";
  curr_category_container→content_description_english = "Code(s)␣for␣Languages";
```

250. ‘a’ — First Language Code for the Current Text.
1. Sprachencode fr den vorliegenden Text.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

[LDF 2006.08.03.] Added code for adding a **Database_Command** that references **Subcategory_Container** :: *f_010_AT_a* to *curr_subcategory_container*.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german =
      "1.␣Sprachencode␣fr␣den␣vorliegenden␣Text";
  curr_subcategory_container→content_description_english = "First␣Language␣Code␣for␣the␣Current␣Tex\
      t";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container :: f_010_AT_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
  category_map["010@"] = curr_category_container;
```

251. 011@ — Year of Appearance.
Erscheinungsjahr.
[LDF 2006.07.24.]

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "011@";
  curr_category_container→pica_3_category_id = "1100";
  curr_category_container→content_description_german = "Erscheinungsjahr";
  curr_category_container→content_description_english = "Year␣of␣Appearance";
```

252. 'a' — Year of Appearance (Beginning), Form for Sorting.
'a' Erscheinugsjahr (Beginn), Sortierform.
[LDF 2006.07.24.]

Log

[LDF 2006.08.04.] Added code for adding a **Database_Command** that references **Subcategory_Container** ::*f_011_AT_a* to *curr_subcategory_container*.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "Erscheinugsjahr␣(Beginn),␣Sortierform";
  curr_subcategory_container→content_description_english = "Year␣of␣Appearance␣(Beginning),␣Form␣fo\
      r␣Sorting";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container ::f_011_AT_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

253. 'b' — Year of Appearance (End), Form for Sorting.
'b' Erscheinugsjahr (Ende), Sortierform.
[LDF 2006.07.24.]

Log

[LDF 2006.08.04.] Added code for adding a **Database_Command** that references **Subcategory_Container** ::*f_011_AT_b* to *curr_subcategory_container*.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'b';
  curr_subcategory_container→content_description_german = "Erscheinugsjahr␣(Ende),␣Sortierform";
  curr_subcategory_container→content_description_english = "Year␣of␣Appearance␣(End),␣Form␣for␣Sort\
      ing";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container ::f_011_AT_b;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['b'] = curr_subcategory_container;
```

254. 'e' — Original Year of Appearance.
Ursprüngliches Erscheinugsjahr.
[LDF 2006.07.24.]

Log

[LDF 2006.08.04.] Added code for adding a **Database_Command** that references **Subcategory_Container** ::*f_011_AT_e* to *curr_subcategory_container*.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'e';
  curr_subcategory_container→content_description_german = "Urspruengliches␣Erscheinugsjahr";
  curr_subcategory_container→content_description_english = "Original␣Year␣of␣Appearance";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container ::f_011_AT_e;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['e'] = curr_subcategory_container;
```

255. 'n' — Year of Appearance (according to RAK-WB).
Erscheinugsjahr (nach RAK-WB).
[LDF 2006.07.24.]

Log

[LDF 2006.08.04.] Added code for adding a **Database_Command** that references **Subcategory_Container** ::*f_011_AT_n* to *curr_subcategory_container*.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'n';
  curr_subcategory_container→content_description_german = "Erscheinugsjahr␣(nach␣RAK-WB)";
  curr_subcategory_container→content_description_english = "Year␣of␣Appearance␣(according␣to␣RAK-WB\
      )";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container ::f_011_AT_n;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['n'] = curr_subcategory_container;
  category_map["011@"] = curr_category_container;
```

256. 016H — General Material Name.
Allgemeine Materialbenennung.
Pica+ 016H, Pica3 1108 [LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "016H";
  curr_category_container→pica_3_category_id = "1108";
  curr_category_container→content_description_german = "Allgemeine␣Materialbenennung";
  curr_category_container→content_description_english = "General␣Material␣Name";
```

257. '0' — General Material Name.
Allgemeine Materialbenennung.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

[LDF 2006.09.01.] Added code that references the function **Subcategory_Container**::*f_016H_0*.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '0';
  curr_subcategory_container→content_description_german = "Allgemeine␣Materialbenennung";
  curr_subcategory_container→content_description_english = "General␣Material␣Name";
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_016H_0;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
```

258.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  category_map["016H"] = curr_category_container;
```

259. 019@ — Code for Country of Publication.
Code für Erscheinungsland.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "019@";
  curr_category_container→pica_3_category_id = "1700";
  curr_category_container→content_description_german = "Code␣fuer␣Erscheinungsland";
  curr_category_container→content_description_english = "Code␣for␣Country␣of␣Publication";
```

260. 'a' — Code for First Country of Publication.
Code für 1. Erscheinungsland.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "Code␣fuer␣1.␣Erscheinungsland";
  curr_subcategory_container→content_description_english = "Code␣for␣First␣Country␣of␣Publication";
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
  category_map["019@"] = curr_category_container;
```

261. 021A — Main Canonical Title, Additional Information, Authorship.
021A — Main Canonical Title, Additional Information, Authorship.
Hauptsachtitel, Zusätze, Verfasserangabe.
Pica+ 021A / Pica3 4000.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "021A";
  curr_category_container→pica_3_category_id = "4000";
  curr_category_container→content_description_german = "Hauptsachtitel,␣Zusaetze,␣Verfasseranga\
      be";
  curr_category_container→content_description_english = "Main␣Canonical␣Title,␣Additional␣Inform\
      ation,␣Authorship";
```

262. ‘1’ — Standard Text.
Standardtext.
[LDF 2006.07.27.]

This field isn’t present in Pica+ 021A / Pica3 4000, but it is present in other categories for titles. It’s used for “j-sets” (“j-Sätze”). I have therefore added it to this category, in order to be able to use a single function to handle multiple categories. [LDF 2006.07.27.]

Log

[LDF 2006.07.27.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '1';
  curr_subcategory_container→content_description_german = "Standardtext";
  curr_subcategory_container→content_description_english = "Standard␣Text";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_021A_1;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['1'] = curr_subcategory_container;
```

263. ‘a’ — Main Canonical Title.
Hauptsachtitel.
[LDF Undated.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "Hauptsachtitel";
  curr_subcategory_container→content_description_english = "Main␣Canonical␣Title";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_021A_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

264. ‘d’ — Additions to the Main Canonical Title.
Zusätze zum Hauptsachtitel.
[LDF Undated.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'd';
  curr_subcategory_container→content_description_german = "Zusaetze␣zum␣Hauptsachtitel";
  curr_subcategory_container→content_description_english = "Additions␣to␣the␣Main␣Canonical␣Title";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_021A_d;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['d'] = curr_subcategory_container;
```

265. 'e' — Additional Creator.
Zu ergänzender Urheber.
[LDF 2006.07.27.]

To Do

[LDF 2006.07.27.] Find a better English translation.

Log

[LDF 2006.07.27.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'e';
  curr_subcategory_container→content_description_german = "Zu␣ergaenzender␣Urheber";
  curr_subcategory_container→content_description_english = "Additional␣Creator";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_021A_e;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['e'] = curr_subcategory_container;
```

266. 'f' — Parallel Canonical Title.
Parallelsachtitel.
[LDF 2006.07.27.]

Log

[LDF 2006.07.27.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'f';
  curr_subcategory_container→content_description_german = "Parallelsachtitel";
  curr_subcategory_container→content_description_english = "Parallel␣Canonical␣Title";
  curr_category_container→subcategory_map['f'] = curr_subcategory_container;
```

267. 'h' Authorship.
Verfasserangabe.
[LDF Undated.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'h';
  curr_subcategory_container→content_description_german = "Verfasserangabe";
  curr_subcategory_container→content_description_english = "Authorship";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_021A_h;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['h'] = curr_subcategory_container;
```

268.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container :: F_021A;
  curr_category_container→database_commands.push_back(curr_database_command);
  category_map["021A"] = curr_category_container;
```

269. 028A — First author.
028A First author.
(1. Verfasser).
Pica+ 028A / Pica3 3000.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "028A";
  curr_category_container→pica_3_category_id = "3000";
  curr_category_container→content_description_german = "1.␣Verfasser";
  curr_category_container→content_description_english = "First␣Author";
```

270. '9' — Identification Number (PPN).
Identifikationsnummer (PPN).
[LDF Undated.]

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '9';
  curr_subcategory_container→content_description_german = "Identifikationsnummer␣(PPN)";
  curr_subcategory_container→content_description_english = "Identification␣Number␣(PPN)";
  curr_category_container→subcategory_map['9'] = curr_subcategory_container;
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container :: f_028A_9;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
```

271. 'a' — Surname.
Familienname.
[LDF Undated.]

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "Familienname";
  curr_subcategory_container→content_description_english = "Surname";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container :: f_028A_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

272. 'c' — Prefix.
Präfix.
[LDF Undated.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'c';
  curr_subcategory_container→content_description_german = "Praefix";
  curr_subcategory_container→content_description_english = "Prefix";
  curr_category_container→subcategory_map['c'] = curr_subcategory_container;
```

273. 'd' — Given Name.
'd' — Vorname.
[LDF Undated.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'd';
  curr_subcategory_container→content_description_german = "Vorname";
  curr_subcategory_container→content_description_english = "Given␣Name";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_028A_d;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['d'] = curr_subcategory_container;
```

274.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_028A;
  curr_category_container→database_commands.push_back(curr_database_command);
  category_map["028A"] = curr_category_container;
```

275. 028B — Personal Name / Author.

Pica+ 028B / Pica3 120, 3000–3009.

Pica3 120:	Personal Name (Cataloguing form according to RSWK) (RSWK = Regeln für den Schlagwortkatalog = Rules for the Subject Catalogue)
Pica3 3000–3009:	Author
Pica3 120:	Personenname (Ansetzungsform nach RSWK) (RSWK = Regeln für den Schlagwortkatalog)
Pica3 3000–3009:	Verfasser

[LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "028B";
  curr_category_container→pica_3_category_id = "120,␣3000--3009";
  curr_category_container→content_description_german = "Personenname␣(Ansetzungsform␣nach␣RSWK)\
      /Verfasser";
  curr_category_container→content_description_german += "";
  curr_category_container→content_description_english = "Personal␣Name␣(Cataloguing␣form␣accordi\
      ng␣to␣RSWK)/Author";
  curr_category_container→content_description_english += "";
```

276. '9' — Identification Number (PPN).

'9' — Identifikationsnummer (PPN).
[LDF 2006.09.07.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '9';
  curr_subcategory_container→content_description_german = "Identifikationsnummer␣(PPN)";
  curr_subcategory_container→content_description_english = "Identification␣Number␣(PPN)";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_028B_9;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['9'] = curr_subcategory_container;
```

277. 'a' — Surname.

'a' — Familienname.
[LDF 2006.09.07.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "Familienname";
  curr_subcategory_container→content_description_english = "Surname";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_028B_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

278. 'd' — Given Name.
'd' — Vorname.
[LDF 2006.09.07.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'd';
  curr_subcategory_container→content_description_german = "Vorname";
  curr_subcategory_container→content_description_english = "Given␣Name";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_028B_d;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['d'] = curr_subcategory_container;
```

279.

Log

[LDF 2006.09.07.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_028B;
  curr_category_container→database_commands.push_back(curr_database_command);
  category_map["028B"] = curr_category_container;
```

280. 028C — Other Contributing Persons.
Sonstige beteiligte Personen.
Pica+ 028C / Pica3 301x/302x.
[LDF Undated.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "028C";
  curr_category_container→pica_3_category_id = "301x/302x";
  curr_category_container→content_description_german = "Sonstige␣beteiligte␣Personen";
  curr_category_container→content_description_english = "Other␣Contributing␣Persons";
```

281. '9' — Identification Number (PPN).
'9' — Identifikationsnummer (PPN).
[LDF Undated.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '9';
  curr_subcategory_container→content_description_german = "Identifikationsnummer␣(PPN)";
  curr_subcategory_container→content_description_english = "Identification␣Number␣(PPN)";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_028C_9;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['9'] = curr_subcategory_container;
```

282. 'a' — Surname.
'a' — Familienname.
[LDF Undated.]

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "Familienname";
  curr_subcategory_container→content_description_english = "Surname";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_028C_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

283. 'c' — Prefix.
'c' — Präfix.
[LDF Undated.]

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'c';
  curr_subcategory_container→content_description_german = "Praefix";
  curr_subcategory_container→content_description_english = "Prefix";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_028C_c;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['c'] = curr_subcategory_container;
```

284. 'd' — Given Name.
'd' — Vorname.
[LDF Undated.]

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'd';
  curr_subcategory_container→content_description_german = "Vorname";
  curr_subcategory_container→content_description_english = "Given␣Name";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_028C_d;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['d'] = curr_subcategory_container;
```

285.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_028C;
  curr_category_container→database_commands.push_back(curr_database_command);
  category_map["028C"] = curr_category_container;
```

286. 033A — Place of Publication, Publisher. Ort, Verlag.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "033A";
  curr_category_container→pica_3_category_id = "4030";
  curr_category_container→content_description_german = "Ort,␣Verlag";
  curr_category_container→content_description_english = "Place␣of␣Publication,␣Publisher";
```

287. 'n' — Publisher.
'n' — Verlag.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'n';
  curr_subcategory_container→content_description_german = "Verlag";
  curr_subcategory_container→content_description_english = "Publisher";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_033A_n;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['n'] = curr_subcategory_container;
```

288. 'p' — Place of Publication.
'p' — Ort.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'p';
  curr_subcategory_container→content_description_german = "Ort";
  curr_subcategory_container→content_description_english = "Place␣of␣Publication";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_033A_p;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['p'] = curr_subcategory_container;
```

289.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_033A;
  curr_category_container→database_commands.push_back(curr_database_command);
  category_map["033A"] = curr_category_container;
```

290. 034D — Size or Range, Specification of Material, Technical System.
034D — Umfangsangabe, spezifische Materialbenennung, technisches System.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "034D";
  curr_category_container→pica_3_category_id = "4060";
  curr_category_container→content_description_german = "Umfangsangabe,␣spezifische␣Materialbene\
      nnung,␣technisches␣System";
  curr_category_container→content_description_english = "Size␣or␣Range,␣Specification␣of␣Materia\
      l,␣Technical␣System";
```

291. 'a' — Text.
[LDF 2006.07.24.]

Log

[LDF 2006.07.24.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "Text";
  curr_subcategory_container→content_description_english = "Text";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_034D_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

292.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  category_map["034D"] = curr_category_container;
```

293. Link To Superordinate Entity. Pica+ 039B / Pica3 4241. [LDF 2006.11.23.]

Log

[2006.11.23.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "039B";
  curr_category_container→pica_3_category_id = "4241";
  curr_category_container→content_description_german = "Titel␣der␣groesseren␣Einheit.";
  curr_category_container→content_description_english = "Link␣To␣Superordinate␣Entity";
```

294. 'a' — Title of the Superordinate Entity.
'a' — Titel der größeren Einheit.
[LDF 2006.11.23.]

Log

[2006.11.23.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "Titel␣der␣groesseren␣Einheit";
  curr_subcategory_container→content_description_english = "Title␣of␣the␣Superordinate␣Entity";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_039B_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

295.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_039B;
  curr_category_container→database_commands.push_back(curr_database_command);
  category_map["039B"] = curr_category_container;
```

296. 041A — Subjects.
041A — Schlagwörter.
Pica+ 041A / Pica3 800, 5100–5199.
[LDF 2006.08.18.]

800:	Main Subject and Subsidiary Subjects (Subject Assignment)
5100–5199:	RSWK Chains
	(RSWK: Regeln für den Schlagwortkatalog = Rules for the Subject Catalogue)

800:	Hauptschlagwort und Unterschlagwörter (Schlagwortansetzung)
5100–5199:	RSWK-Ketten
	(RSWK: Regeln für den Schlagwortkatalog)

[LDF 2006.08.18.]

Log

[LDF 2006.08.18.] Added this section.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "041A";
  curr_category_container→pica_3_category_id = "800,␣5100--5199";
  curr_category_container→content_description_german = "Hauptschlagwort␣und␣Unterschlagwoerter␣\
      (Schlagwortansetzung)/";
  curr_category_container→content_description_german += "RSWK-Ketten";
  curr_category_container→content_description_english = "Main␣Subject␣and␣Subsidiary␣Subjects␣(S\
      ubject␣Assignment)/";
  curr_category_container→content_description_english += "RSWK␣Chains";
```

297. '9' — Identification Number (PPN).
'9' — Identifikationsnummer (PPN).
[LDF 2006.08.21.]
This field is only present in Pica3 5100–5199, not Pica3 800. [LDF 2006.08.21.]

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '9';
  curr_subcategory_container→content_description_german = "Identifikationsnummer␣(PPN)";
  curr_subcategory_container→content_description_english = "Identification␣Number␣(PPN)";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container :: f_041A_9;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['9'] = curr_subcategory_container;
```

298. 'a' — Subject.
'a' — Main or Subsidiary Subject/Subject; Second and additional permutation pattern; Subject Chain Information.
'a' — Hauptschlagwort/Schlagwort; Zweites und weiteres Permutationsmuster; Angaben zur Schlagwortkette.
[LDF 2006.08.18.]

This field is called "Haupt- bzw. Unterschlagwort" ("Main or Subsidiary Subject") in Pica3 800, and "Schlagwort" ("Subject") in Pica3 5100–5199. In Pica3 5100–5199, it is used for two other purposes:

1. "Second and additional permutation pattern" ("zweites und weiteres Permutationsmuster"), when preceded by the field 'f' (Permutationsmuster/Permutation Pattern).
2. "Subject Chain Information" ("Angaben zur Schlagwortkette").

This unfortunate ambiguity makes it much more complicated to process entries where Category 041A, Field 'a', and possibly Field 'f' are present. [LDF 2006.08.21.]

Log

[LDF 2006.08.18.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german =
      "Hauptschlagwort/Schlagwort;␣␣Zweites␣und␣weiteres␣Permutationsmuster;␣␣";
  curr_subcategory_container→content_description_german += "Angaben␣zur␣Schlagwortkette";
  curr_subcategory_container→content_description_english = "Main␣Subject/Subject;␣␣Second␣and␣addit\
      ional␣permutation␣pattern;␣␣";
  curr_subcategory_container→content_description_english += "Subject␣Chain␣Information";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_041A_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

299. 'f' — Permutation Pattern.
'f' — Permutationsmuster.
[LDF 2006.08.21.]

Log

[LDF 2006.08.21.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'f';
  curr_subcategory_container→content_description_german = "Permutationsmuster";
  curr_subcategory_container→content_description_english = "Permutation␣Pattern";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_041A_f;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['f'] = curr_subcategory_container;
```

300. 'S' — Indicator/Subject Indicator.
'S' — Indikator/Schlagwortindikator.
[LDF 2006.08.21.]

This field is called "Indikator" ("Indicator") in Pica3 800 and "Schlagwortindikator" ("Subject Indicator") in Pica3 5100–5199. [LDF 2006.08.21.]

Log

[LDF 2006.08.18.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
   curr_subcategory_container = new Subcategory_Container;
   curr_subcategory_container→pica_plus_field_id = 'S';
   curr_subcategory_container→content_description_german = "Indikator/SchlagwortIndikator";
   curr_subcategory_container→content_description_english = "Indicator/Subject␣Indicator";
   curr_database_command = new Database_Command;
   curr_database_command→function = Subcategory_Container::f_041A_S;
   curr_subcategory_container→database_commands.push_back(curr_database_command);
   curr_category_container→subcategory_map['S'] = curr_subcategory_container;
```

301.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
   curr_database_command = new Database_Command;
   curr_database_command→function = Category_Container::F_041A;
   curr_category_container→database_commands.push_back(curr_database_command);
   category_map["041A"] = curr_category_container;
```

302. 047I — Content Summary (short).
047I — Inhaltliche Zusammenfassung (kurz).
Pica+ 047I / Pica3 4207.
[LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
   curr_category_container = new Category_Container;
   curr_category_container→pica_plus_category_id = "047I";
   curr_category_container→pica_3_category_id = "4207";
   curr_category_container→content_description_german = "Inhaltliche␣Zusammenfassung␣(kurz)";
   curr_category_container→content_description_english = "Content␣Summary␣(short)";
```

303. 'a' — Text.
[LDF 2006.09.07.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "Text";
  curr_subcategory_container→content_description_english = "Text";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_047I_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

304.

Log

[LDF 2006.09.07.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
#if 0      /* 1 */
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_047I;
  curr_category_container→database_commands.push_back(curr_database_command);
#endif
  category_map["047I"] = curr_category_container;
```

305. 203@ — Exemplar Production Number.
203@ — Exemplar-Produktionsnummer.
Pica+ 203@ / Pica3 7800.
[LDF 2006.08.17.]

Log

[LDF 2006.08.17.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "203@";
  curr_category_container→pica_3_category_id = "7800";
  curr_category_container→content_description_german = "Exemplar-Produktionsnummer";
  curr_category_container→content_description_english = "Exemplar␣Production␣Number";
```

306. ‘0’ — Exemplar Production Number.
‘0’ — Exemplar-Produktionsnummer.
[LDF 2006.08.17.]

Log

[LDF 2006.08.17.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '0';
  curr_subcategory_container→content_description_german = "Exemplar-Produktionsnummer";
  curr_subcategory_container→content_description_english = "Exemplar␣Production␣Number";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_203_AT_0;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
```

307. Category_Container::F_203_AT not needed at present. [LDF 2006.08.17.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
#if 0
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_203_AT;
  curr_category_container→database_commands.push_back(curr_database_command);
#endif
  category_map["203@"] = curr_category_container;
```

308. 209A — Call Number.
209A — Signatur.
Pica+ 209A / Pica3 7100–7109.
[LDF 2006.08.17.]

Log

[LDF 2006.08.17.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "209A";
  curr_category_container→pica_3_category_id = "7100--7109";
  curr_category_container→content_description_german = "Signatur";
  curr_category_container→content_description_english = "Call␣Number";
```

309. 'a' — Call Number.
'a' — Signatur.
[LDF 2006.08.17.]

Log

[LDF 2006.08.17.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "Signatur";
  curr_subcategory_container→content_description_english = "Call␣Number";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209A_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

310. 'b' — Library Number.
'b' — Bibliotheksnummer.
[LDF 2006.08.17.]

Log

[LDF 2006.08.17.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'b';
  curr_subcategory_container→content_description_german = "Bibliotheksnummer";
  curr_subcategory_container→content_description_english = "Library␣Number";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209A_b;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['b'] = curr_subcategory_container;
```

311. 'f' — Special Location.
'f' — Sonderstandort.
[LDF 2006.08.17.]

Log

[LDF 2006.08.17.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'f';
  curr_subcategory_container→content_description_german = "Sonderstandort";
  curr_subcategory_container→content_description_english = "Special␣Location";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209A_f;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['f'] = curr_subcategory_container;
```

312. 'j' — Library Department.
'j' — Abteilung der Bibliothek.
[LDF 2006.08.17.]

Log

[LDF 2006.08.17.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'j';
  curr_subcategory_container→content_description_german = "Abteilung␣der␣Bibliothek";
  curr_subcategory_container→content_description_english = "Library␣Department";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209A_j;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['j'] = curr_subcategory_container;
```

313.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_209A;
  curr_category_container→database_commands.push_back(curr_database_command);
  category_map["209A"] = curr_category_container;
```

314. 209C — Access Number.
209C — Zugangsnummer.
Pica+ 209C / Pica3 8100
[LDF 2006.08.18.]

Log

[LDF 2006.08.18.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "209C";
  curr_category_container→pica_3_category_id = "8100";
  curr_category_container→content_description_german = "Zugangsnummer";
  curr_category_container→content_description_english = "Access␣Number";
```

315. ‘a’ — Access Number.
‘a’ — Zugangsnummer.
[LDF 2006.08.18.]

Log

[LDF 2006.08.18.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "Zugangsnummer";
  curr_subcategory_container→content_description_english = "Access␣Number";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209C_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

316.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
#if 0
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_209C;
  curr_category_container→database_commands.push_back(curr_database_command);
#endif
  category_map["209C"] = curr_category_container;
```

317. 209R — Remote Access.
Local information regarding remote access to electronic resources.
Lokale Angaben zum Zugriff auf elektronische Ressourcen im Fernzugriff.
Pica+ 209R / Pica3 7133.
[LDF 2006.08.15.]

Log

[LDF 2006.08.15.] Added this section.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_category_container = new Category_Container;
  curr_category_container→pica_plus_category_id = "209R";
  curr_category_container→pica_3_category_id = "7133";
  curr_category_container→content_description_german = "Lokale␣Angaben␣zum␣Zugriff␣auf␣elektron\
      ische␣Ressourcen␣im␣Fernzugriff";
  curr_category_container→content_description_english = "Local␣information␣regarding␣remote␣acce\
      ss␣to␣electronic␣resources";
```

318. ‘0’ — Format.
[LDF 2006.08.16.]

Log

[LDF 2006.08.16.] Added this section.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = '0';
  curr_subcategory_container→content_description_german = "Format";
  curr_subcategory_container→content_description_english = "Format";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container :: f_209R_0;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['0'] = curr_subcategory_container;
```

319. ‘a’ — URL (Universal Resource Locator).
[LDF 2006.08.16.]

Log

[LDF 2006.08.16.] Added this section.

```
⟨ Define ZClient :: init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'a';
  curr_subcategory_container→content_description_german = "URL␣(Universal␣Resource␣Locator)";
  curr_subcategory_container→content_description_english = "URL␣(Universal␣Resource␣Locator)";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container :: f_209R_a;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['a'] = curr_subcategory_container;
```

320. 'g' — URN (Universal Resource Name). 'g' — URN (Universal Resource Name).
[LDF 2006.08.16.]

Log

[LDF 2006.08.16.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'g';
  curr_subcategory_container→content_description_german = "URN␣(Universal␣Resource␣Name)";
  curr_subcategory_container→content_description_english = "URN␣(Universal␣Resource␣Name)";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209R_g;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['g'] = curr_subcategory_container;
```

321. 'S' — License indicator.
'S' — Lizenzindikator.
[LDF 2006.08.15.]

Log

[LDF 2006.08.15.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'S';
  curr_subcategory_container→content_description_german = "Lizenzindikator";
  curr_subcategory_container→content_description_english = "License␣indicator";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209R_S;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['S'] = curr_subcategory_container;
```

322. 'x' — Internal Remarks.
'x' — Interne Bemerkungen.
[LDF 2006.08.16.]

Log

[LDF 2006.08.16.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'x';
  curr_subcategory_container→content_description_german = "Interne␣Bemerkungen";
  curr_subcategory_container→content_description_english = "Internal␣Remarks";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209R_x;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['x'] = curr_subcategory_container;
```

323. 'y' — Text for the Web Display.
'y' — Text für die Web-Anzeige.
[LDF 2006.08.16.]

Log

[LDF 2006.08.16.] Added this section.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_subcategory_container = new Subcategory_Container;
  curr_subcategory_container→pica_plus_field_id = 'y';
  curr_subcategory_container→content_description_german = "Text␣fuer␣die␣Web-Anzeige";
  curr_subcategory_container→content_description_english = "Text␣for␣the␣Web␣Display";
  curr_database_command = new Database_Command;
  curr_database_command→function = Subcategory_Container::f_209R_y;
  curr_subcategory_container→database_commands.push_back(curr_database_command);
  curr_category_container→subcategory_map['y'] = curr_subcategory_container;
```

324.

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  curr_database_command = new Database_Command;
  curr_database_command→function = Category_Container::F_209R;
  curr_category_container→database_commands.push_back(curr_database_command);
  category_map["209R"] = curr_category_container;
```

325. Show *category_map*. [LDF Undated.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
#if 0
  for (Category_Map_Type::iterator iter = category_map.begin(); iter ≠ category_map.end();
        ++iter) {
    temp_strm ≪ "category_map[" ≪ iter→first ≪ "]:\n";
    iter→second→show(temp_strm);
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
  }
#endif
```

326. End of **ZClient**::*init_category_map* definition. [LDF Undated.]

```
⟨ Define ZClient::init_category_map 225 ⟩ +≡
  return 0; }     /* End of ZClient::init_category_map definition. */
```

327. Search using PQF syntax. [LDF Undated.]
"PQF" stands for "Prefix Query Syntax". See the *YAZ User's Guide and Reference* (`http://www.indexdata.com/yaz/doc/tools.tkl#PQF`) for more information. [LDF 2006.10.11.]

```
⟨ Declare ZClient functions 197 ⟩ +≡
  int search_pqf(char *curr_query_str, ZOOM_connection *curr_connection = 0);
```

328.

```
⟨ Define ZClient functions 198 ⟩ +≡
  int ZClient::search_pqf(char *curr_query_str, ZOOM_connection *curr_connection)
  {
    stringstream temp_strm;
    resultsets.push_back(static_cast⟨ZOOM_resultset *⟩(malloc(sizeof(ZOOM_resultset))));
    ZOOM_resultset *curr_resultset = resultsets.back( );
    if (curr_connection ≡ 0 ∧ connections.back( ) ≠ 0) curr_connection = connections.back( );
    else if (curr_connection ≡ 0) {
      temp_strm ≪ "ERROR!␣␣In␣‘ZClient::search_pqf’:" ≪
          endl ≪ "‘curr_connection’␣argument␣is␣0␣and␣" ≪
          "‘vector<ZOOM_connection*>␣this->connections’␣" ≪ "is␣empty." ≪
          endl ≪ "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
      return 1;
    }    /* else if (curr_connection ≡ 0) */
    *curr_resultset = ZOOM_connection_search_pqf(*curr_connection, curr_query_str);
    return 0;
  }    /* End of ZClient::search_pqf definition. */
```

329. Parse Records. [LDF Undated.]

Log

[LDF 2006.09.06.] Changed name of **ofstream** *log_file* data member to *log_strm*.

[LDF 2006.09.06.] Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

```
⟨ Declare ZClient functions 197 ⟩ +≡
  int parse_records(const char *in_filename);
```

330.

```
⟨ Define ZClient functions 198 ⟩ +≡
  int ZClient::parse_records(const char *in_filename){ stringstream temp_strm;
#if 0    /* 1 */
      temp_strm ≪ "Entering␣‘ZClient::parse_records’.";
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
#endif
      int result = 0;
```

331. Open the file containing the records for input and a log file for output. The name of the input file is passed to this function as an argument. In this example, it's `record.txt`. The log file is `log.txt`. [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  ifstream in_file;
  in_file.open(in_filename);
```

332. We'll be reading characters one-by-one from the input file. Sometimes we'll need one character of "lookahead", hence *next_char*. The characters must be **unsigned**, because the input file contains special characters, which would otherwise be interpreted as negative numbers, causing a run-time error. [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  unsigned char curr_char = '\0';
  unsigned char next_char = '\0';
```

333. The fields in PICA are marked by the character °*237* followed by another character. *curr_field_id* and *prev_field_id* are used to store this second character. They are initialized to `'\0'`, which is just a name for 0 that makes it clear that it's a character rather than some other kind of number. [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  unsigned char curr_field_id = '\0';
  unsigned char prev_field_id = '\0';
```

334. The number of characters read. Not currently used for anything. [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  unsigned long char_ctr = 0;
```

335. The number of metadata records processed. [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  unsigned long record_ctr = 0;
```

336. PICA+ records have "categories", with codes such as "001@", "027A", "028/03", etc., and "fields" with codes as described above. [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  string curr_category_id;
  string curr_field_value;
  string curr_repeat_code;
```

337. The variable *state* is used to store information about what the program is doing at a given moment. The state of the program determines how characters are interpreted. [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  const unsigned short NULL_STATE = 0;
  const unsigned short OUTSIDE_RECORD = 1;
  const unsigned short COLLECTING_CATEGORY_ID = 2;
  const unsigned short COLLECTING_FIELD = 3;
  unsigned short state = OUTSIDE_RECORD;
#if 0
  temp_strm ≪ "About␣to␣parse␣records.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  Pica_Record *curr_pica_record = 0; while (true) { curr_char = in_file.get();
    /* Read a character from the input file. LDF 2006.07.18. */
  ++char_ctr;
```

338. This is EOF (end-of-file). It would normally be -1, but *curr_char* is unsigned. When we read this, we're done. Some error handling should be added here. [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  if (curr_char ≡ 255) {
    if (curr_pica_record ≠ 0) {
      result = curr_pica_record→write_to_database(this, database, log_strm);
      temp_strm ≪ "‘curr_pica_record->write_to_database’␣returned␣==␣" ≪ result ≪ endl;
      curr_pica_record→show(temp_strm, "curr_pica_record:");
      log_strm.lock( );
      log_strm.output_stream ≪ temp_strm.str( ) ≪ endl ≪ endl;
      log_strm.unlock( );
#if 0    /* 1 */
      AfxMessageBox(temp_strm.str( ).c_str( ));
#endif
      temp_strm.str("");
      delete curr_pica_record;
      curr_pica_record = 0;
    }
    break;
  }    /* if (curr_char ≡ 255) (EOF) */
  else if (curr_char ≡ '\n')    /* Newline. */
  {
```

339. Not reading a record, so just keep going. [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  if (state ≡ OUTSIDE_RECORD) {
    continue;
  }
```

340. A line in a record starts with category id, contains a variable number of field ids with associated text, and ends in a newline. So a newline marks the end of the information associated with a particular category id. Some categories can appear more than once within a record, but then the category id must be repeated at the beginning of another line. [LDF 2006.07.18.]

Log

[LDF 2006.07.20.] !! BUG FIX: Now resetting *prev_field_id* to '\0'.

```
⟨ Define ZClient functions 198 ⟩ +≡
  else if (state ≡ COLLECTING_FIELD) { prev_field_id = '\0';    /* Reset prev_field_id */
  temp_strm ≪ "Finished␣collecting␣‘curr_field_value:" ≪ endl ≪
      "curr_field_value␣==␣" ≪ curr_field_value ≪ endl ≪ "‘curr_field_id’␣==␣" ≪
      curr_field_id ≪ endl ≪ "‘prev_field_id’␣==␣" ≪ prev_field_id ≪ endl ≪
      "Setting␣‘state’␣to␣‘COLLECTING_CATEGORY_ID’.";
  log_strm.lock( );
  log_strm.output_stream ≪ "\n\n␣***␣" ≪ temp_strm.str( ) ≪ "\n\n" ≪ curr_field_value ≪ "␣";
  log_strm.unlock( );
#if 0
  AfxMessageBox(temp_strm.str( ).c_str( ));
#endif
  temp_strm.str("");
```

341. This is one of two places where we must process the information we've gathered. It's put into **Pica_Record** *curr_pica_record*. [LDF 2006.07.20.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  result = write_field_data(curr_category_id, curr_repeat_code, curr_field_id, curr_field_value, curr_pica_record);
```

342. Peek at the next character without removing it from the input stream. Actually, "peeking" does remove it, but puts it back right away. If we have two newlines in a row, that's the end of the record. In this case, when the second newline is read again, *state* will be `OUTSIDE_RECORD`, so it will be ignored. If *next_char* isn't a newline, there should be more lines with information, or it might be EOF. [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  next_char = in_file.peek();
  if (next_char ≡ '\n') {
    temp_strm ≪ "End␣of␣record␣" ≪ record_ctr ≪ endl ≪
        "Setting␣'state'␣to␣'OUTSIDE_RECORD'.";
    log_strm.lock();
    log_strm.output_stream ≪ "\n\n␣***␣" ≪ temp_strm.str() ≪ "\n\n";
    log_strm.unlock();
#if 0     /* 1 */
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    if (curr_pica_record ≠ 0) {
      curr_pica_record→write_to_database(this, database, log_strm);
      curr_pica_record→show(temp_strm, "curr_pica_record:");
      log_strm.lock();
      log_strm.output_stream ≪ temp_strm.str() ≪ endl ≪ endl;
      log_strm.unlock();
#if 0     /* 1 */
      AfxMessageBox(temp_strm.str().c_str());
#endif
      temp_strm.str("");
      delete curr_pica_record;
      curr_pica_record = 0;
    }
    state = OUTSIDE_RECORD;
  }    /* if (next_char ≡ '\n') */
  else {
    state = COLLECTING_CATEGORY_ID;
  }
```

343. Reset the *strings* and *chars* we're using to gather information to the empty string and the null character, respectively. [LDF 2006.07.18.]

Log

[LDF 2006.07.20.] Now setting *curr_repeat_code* to "".

```
⟨ Define ZClient functions 198 ⟩ +≡
  curr_category_id = "";
  curr_repeat_code = "";
  curr_field_value = "";
  prev_field_id = curr_field_id;
  curr_field_id = '\0'; }    /* else if (state ≡ COLLECTING_FIELD) */
```

344. This was just for testing purposes. [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  else {
    log_strm.lock();
    log_strm.output_stream ≪ "[NEWLINE]" ≪ curr_char;
    log_strm.unlock();
  }
  }    /* else if (curr_char ≡ '\n') */
```

345. °237: Control sequence for a field.

```
⟨ Define ZClient functions 198 ⟩ +≡
  else if (curr_char ≡ °237) { curr_char = in_file.get();
  ++char_ctr;
  prev_field_id = curr_field_id;
  curr_field_id = curr_char;
```

346. Handle current field, if it's not empty. We've got a new field id, so if there was an old one, we need to handle it, and its associated text, before we start collecting the text for the new one. This is the second place where information will be written to a database (probably). [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  if (prev_field_id ≠ '\0') {
    temp_strm ≪ "Got␣a␣control␣sequence␣for␣a␣field,␣and␣|prev_field_id|␣" ≪
        "is␣non-null." ≪ endl ≪ "‘prev_field_id’␣==␣" ≪ prev_field_id ≪ endl ≪
        "‘curr_field_value’␣==␣" ≪ curr_field_value ≪ endl ≪ "‘curr_char_id’␣==␣" ≪
        curr_char ≪ endl ≪ "‘curr_category_id’␣==␣" ≪ curr_category_id;
    log_strm.lock();
    log_strm.output_stream ≪ "\n\n***␣" ≪ temp_strm.str() ≪ "\n\n";
    log_strm.unlock();
#if 0
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    result = write_field_data(curr_category_id, curr_repeat_code, prev_field_id, curr_field_value,
        curr_pica_record);
    curr_field_value = "";    /* Reset curr_field_value */
  }    /* if (prev_field_id ≠ '\0') */
```

347. The first field id for a line. [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  else     /* prev_field_id ≡ '\0' */
  {
    temp_strm ≪ "Got␣a␣control␣sequence␣for␣a␣field,␣and␣|prev_field_id|␣" ≪ "is␣null." ≪
        endl ≪ "Not␣doing␣anything." ≪ endl ≪ "`curr_category_id'␣==␣" ≪ curr_category_id ≪
        endl ≪ "`curr_field_id␣==␣" ≪ curr_field_id ≪ endl ≪ "`curr_field_value'␣==␣" ≪
        curr_field_value;
    log_strm.lock( );
    log_strm.output_stream ≪ "\n\n***␣" ≪ temp_strm.str( ) ≪ "\n\n";
    log_strm.unlock( );
#if 0
    AfxMessageBox(temp_strm.str( ).c_str( ));
#endif
    temp_strm.str("");
    curr_field_value = "";
  }    /* else (prev_field_id ≡ '\0') */
  }    /* else if (curr_char ≡ °237) */
  else if (isspace(curr_char)) {
```

348. Skip over spaces before the first record, or between records. There shouldn't be any, though. [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  if (state ≡ OUTSIDE_RECORD) {
    continue;
  }
```

349. A category id is terminated by a space. [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  else if (state ≡ COLLECTING_CATEGORY_ID) { temp_strm ≪
      "Finished␣collecting␣`curr_category_id':" ≪ endl ≪ "curr_category_id␣==␣" ≪
      curr_category_id ≪ endl ≪ "Setting␣`state'␣to␣`COLLECTING_FIELD'.\n" ≪
      "curr_category_id.length()␣==␣" ≪ static_cast⟨unsigned int⟩(curr_category_id.length( ));
  string::size_type pos;
  if (curr_category_id.length( ) > 4) {
    pos = curr_category_id.find('/');
    curr_repeat_code = curr_category_id.substr(pos + 1);
    curr_category_id.erase(pos);
  }
```

350. Now we start collecting the information for a field. The next thing we should read should be a "field id", followed by a "field value". [LDF 2006.07.18.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  state = COLLECTING_FIELD;
  curr_field_value = ""; }
```

351. Append *curr_char* to *curr_field_value*. The case that *curr_char* is a field id is caught above. It's always the single character that follows °*237*, and it can't be a space. [LDF 2006.07.19.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  else
    if (state ≡ COLLECTING_FIELD) {
      curr_field_value += curr_char;
    }
```

352. Just for testing. [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  else {
    log_strm.lock( );
    log_strm.output_stream ≪ "[SPACE]" ≪ curr_char;
    log_strm.unlock( );
  }
  }     /* else if (isspace(curr_char)) */
```

353. Other characters. [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  else {
```

354. We've found a character, so start collecting a category id. Please note: The following conditional is **if**, not **else if**, so after this code executes, it will be true, and the code it controls will be executed. [LDF 2006.07.19.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  if (state ≡ OUTSIDE_RECORD) {
    state = COLLECTING_CATEGORY_ID;
    ++record_ctr;
    curr_pica_record = new Pica_Record;
  }
```

355. We're collecting a category id, so add this character to it. [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  if (state ≡ COLLECTING_CATEGORY_ID) {
    curr_category_id += curr_char;
  }
```

356. We're collecting the text for a field, so add this character to it. The case that *curr_char* is a field id is caught above. It's always the single character that follows °*237*, and it can't be a space. [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
  else
    if (state ≡ COLLECTING_FIELD) {
      curr_field_value += curr_char;
    }
  log_strm.lock( );
  log_strm.output_stream ≪ curr_char;
  log_strm.unlock( ); }     /* else (Other characters.) */
  }     /* while */
```

357. We've read EOF, so we're done. [LDF Undated.]

```
⟨ Define ZClient functions 198 ⟩ +≡
#if 0    /* 1 */
  temp_strm << "ZClient::Exiting␣‘parse_records’.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  in_file.close();
  return 0; }    /* End of ZClient::parse_records definition. */
```

358. Write Field Data. [LDF Undated.]

Log

[LDF 2006.09.06.] Changed name of **ofstream** *log_file* data member to *log_strm*.

[LDF 2006.09.06.] Made **stringstream** *temp_strm* a local variable within **ZClient**::*write_field_data*. Formerly, it was **static** and local to this file. I've made this change in the interests of thread-safety.

[LDF 2006.09.06.] Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

```
⟨ Declare ZClient functions 197 ⟩ +≡
  int write_field_data(
  string &category_id,
  string &repeat_code,
  char field_id,
  string &field_value,
  Pica_Record *pica_record);
```

359.

```
⟨ Define ZClient functions 198 ⟩ +≡
  int ZClient::write_field_data(
  string &category_id,
  string &repeat_code,
  char field_id,
  string &field_value,
  Pica_Record *pica_record){ stringstream temp_strm;
      Category_Container *curr_category_container = 0;
      Category_Container *curr_pica_record_category_container = 0;
      Subcategory_Container *curr_subcategory_container = 0;
      Subcategory_Container *curr_pica_record_subcategory_container = 0;
      curr_category_container = category_map[category_id];
      if (curr_category_container ≡ 0) {
        temp_strm ≪ "In␣‘ZClient::write_field_data’:␣␣Category␣" ≪ category_id ≪
            "␣doesn’t␣exist␣yet.␣␣Ignoring␣and␣returning␣1.";
        log_strm.lock();
        log_strm.output_stream ≪ "\n***␣" ≪ temp_strm.str() ≪ endl ≪ endl;
        log_strm.unlock();
#if 0
        AfxMessageBox(temp_strm.str().c_str());
#endif
        temp_strm.str("");
        return 1;
      }
      else    /* curr_category_container ≠ 0 */
      { temp_strm ≪ "Category␣" ≪ category_id ≪ "␣==␣" ≪
          curr_category_container→content_description_german ≪ endl;
      curr_subcategory_container = curr_category_container→subcategory_map[field_id];
      if (curr_subcategory_container ≡ 0) {
        temp_strm ≪ "In␣‘ZClient::write_field_data’:␣␣Subcategory␣" ≪ field_id ≪
            "␣doesn’t␣exist␣yet.␣␣Returning␣1.";
        log_strm.lock();
        log_strm.output_stream ≪ "\n***␣" ≪ temp_strm.str() ≪ endl ≪ endl;
        log_strm.unlock();
#if 0
        AfxMessageBox(temp_strm.str().c_str());
#endif
        temp_strm.str("");
        return 1;
      }
      else    /* curr_subcategory_container ≠ 0 */
      { temp_strm ≪ "In␣‘ZClient::write_field_data’:␣␣Subcategory␣" ≪ field_id ≪ "␣==␣␣" ≪
          curr_subcategory_container→content_description_german ≪ endl ≪ "‘field_value’␣==␣" ≪
          field_value;
      log_strm.lock();
      log_strm.output_stream ≪ "\n***␣" ≪ temp_strm.str() ≪ endl ≪ endl;
      log_strm.unlock();
#if 0
      AfxMessageBox(temp_strm.str().c_str());
#endif
      temp_strm.str("");
      curr_pica_record_category_container = 0;
      pair⟨Category_Multimap_Type::iterator, Category_Multimap_Type::iterator⟩
          iter_pair = pica_record→categories.equal_range(category_id);
      if (iter_pair.first ≠ pica_record→categories.end() ∧ iter_pair.first→first ≡ category_id) {
        for (Category_Multimap_Type::iterator iter = iter_pair.first; iter ≠ iter_pair.second;
              ++iter) {
          if (iter→second→repeat_code ≡ repeat_code) curr_pica_record_category_container = iter→second;
        }
      }
      if (curr_pica_record_category_container ≡ 0) {
#if 0    /* 1 */
        temp_strm ≪ "In␣‘ZClient::write_field_data’:␣␣" ≪
            "‘curr_pica_record_category_container’␣==␣0" ≪ endl ≪
            "Creating␣a␣new␣‘Category_Container’␣and␣putting␣it␣" ≪
            "onto␣‘pica_record->categories’.";
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
        curr_pica_record_category_container = new Category_Container;
        ∗curr_pica_record_category_container = ∗curr_category_container;
        curr_pica_record_category_container→repeat_code = repeat_code;
        pica_record→categories.insert(make_pair(category_id, curr_pica_record_category_container));
#if 0    /* 1 */
        pica_record→categories[category_id]→show(temp_strm);
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
      }    /* if (curr_pica_record_category_container ≡ 0) */
      else    /* curr_pica_record_category_container ≠ 0 */
      {
#if 0    /* 1 */
        temp_strm ≪ "In␣‘ZClient::write_field_data’:␣␣" ≪
            "‘curr_pica_record_category_container’␣!=␣0" ≪ endl ≪
            "A␣‘Category_Container’␣already␣exists.";
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
      }    /* else (curr_pica_record_category_container ≠ 0) */
```

360. Add **Subcategory_Container** and push it onto *curr_pica_record_category_container*. [LDF Undated.] ▮

```
⟨ Define ZClient functions 198 ⟩ +≡
  curr_pica_record_subcategory_container = new Subcategory_Container;
  *curr_pica_record_subcategory_container = *curr_subcategory_container;
  curr_pica_record_subcategory_container→field_value = field_value;
  curr_pica_record_category_container→subcategory_vector.push_back(make_pair(field_id,
        curr_pica_record_subcategory_container));
#if 0
  temp_strm ≪ "In␣‘ZClient::write_field_data’:␣␣Returning␣0.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  return 0; }     /* else (curr_subcategory_container ≠ 0) */
  }     /* else (curr_category_container ≠ 0) */
  }     /* End of ZClient::write_field_data definition. */
```

361. Putting ZClient together. [LDF 2006.09.18.]

362. This is what's compiled.

```
  ⟨ Include files 13 ⟩
  ⟨ zclient.web 188 ⟩
  ⟨ Initialize static constants for ZClient 194 ⟩
  ⟨ Define ZClient functions 198 ⟩
  ⟨ Define ZClient::init_category_map 225 ⟩
```

363. This is what gets written to `zclient.h`. [LDF Undated.]

⟨ `zclient.h` 363 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
#include `<vector>`
  ⟨ **using** declarations for namespaces 191 ⟩
  ⟨ Declare **class ZClient** 192 ⟩

364. ZOOM functions (`ztstzoom.web`). [LDF 2006.10.05.]
The functions in this file contain code and comments from files from the YAZ distribution. [LDF 2006.10.05.] ▮

Log

Created this file. It contains code formerly in `ldf_zoomtst.h`, `zoomtst0.cpp`, `zoomtst2.cpp`, and `zoomtst3.cpp`.▮ `zoomtst1.cpp` was empty.

Server: `z3950.gbv.de`

Character Set	Port
ISO 5426	210
ISO 8859-1	20010
ANSEL	20011

Guest Login ID: 999
Passwort: abc

⟨ `ztstzoom.web` 364 ⟩ ≡
  static char *id_string*[ ] = `"$Id:␣ztstzoom.web,v␣1.7␣2006/11/28␣16:25:12␣Administrator␣Exp␣$"`;

This code is cited in sections 6 and 8.
This code is used in section 382.

365. Include files.

⟨ Include files 13 ⟩ +≡
#include `"stdafx.h"`

366. using declarations for namespaces.

⟨ **using** declarations for namespaces 191 ⟩ +≡
  using namespace std;

367. zoomtst0. [LDF Undated.]

⟨ Declare ZOOM functions 367 ⟩ ≡
  int *zoomtst0* (**void**);

See also sections 369 and 379.
This code is used in section 383.

368.

```
⟨ Define ZOOM functions 368 ⟩ ≡
  int zoomtst0(void){ const char *errmsg, *addinfo; int error ;
      stringstream temp_strm;    /* GBV */
#if 0    /* 1 */
      ZOOM_connection z = ZOOM_connection_new("z3950.gbv.de:20010/GVK", 20010);
#endif
#define USE_GBV 1    /* 0 */
#if USE_GBV
      ZOOM_connection z = ZOOM_connection_new("z3950.gbv.de:20010/GVK", 0);
#else
      ZOOM_connection z = ZOOM_connection_new("z3950.loc.gov:7090/Voyager", 0);
#endif
      if ( ( error = ZOOM_connection_error(z, &errmsg,
          &addinfo) ) ) { temp_strm ≪ "Error:␣" ≪ errmsg ≪ "␣(" ≪ error ≪ ",␣" ≪ addinfo;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
      exit(2); } ZOOM_resultset r;
      const char *rec;
#if USE_GBV
      ZOOM_connection_option_set(z, "user", "999");
      ZOOM_connection_option_set(z, "password", "abc");
      r = ZOOM_connection_search_pqf(z, "@attr␣1=1␣Finston");
#else
      ZOOM_connection_option_set(z, "preferredRecordSyntax", "USMARC");
      r = ZOOM_connection_search_pqf(z, "@attr␣1=7␣0253333490");
#endif
      rec = ZOOM_record_get(ZOOM_resultset_record(r, 0), "render", 0);
      if (rec) {
        temp_strm ≪ "Record:␣" ≪ rec;
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
      }
      else AfxMessageBox("rec␣is␣NULL");
      return 0; }    /* End of zoomtst0 definition. */
```

See also sections 370, 371, 372, 373, 374, 375, 376, 377, 378, and 380.
This code is used in section 382.

369. zoomtst2. [LDF Undated.]

Log

[LDF Undated.] Added this function.

[LDF 2006.08.31.] Added the optional **bool** arguments *perform_search*, *write_records_file*, *clear_database* , *parse_records*, and **bool** *display_records*, all with the default *true*. Now using these variables rather than preprocessor macros to control whether *zoomtst2* performs a search, writes the file `records.txt`, clears the database, parses the records, and/or displays records.

[LDF 2006.09.25.] Changed the type of the **bool** arguments to **BOOL** and their defaults to `TRUE`. Using **bool** and *true* caused Microsoft Visual Studio to issue warnings.

[LDF 2006.10.16.] Added the argument **unsigned short** *server_selector*.

```
⟨ Declare ZOOM functions 367 ⟩ +≡
  int zoomtst2 (
  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);
```

370.

```
⟨ Define ZOOM functions 368 ⟩ +≡
  int zoomtst2 (
  unsigned short server_selector,
  CString search_command,
  BOOL perform_search,
  BOOL write_records_file,
  BOOL clear_database,
  BOOL parse_records,
  BOOL display_records) { stringstream temp_strm;
      stringstream sql_strm;
      stringstream message_strm;
      ZClient z(ZClient::GBV_GVK_ID, "z3950.gbv.de:20010/GVK");
      z.init_category_map( );
      temp_strm ≪ "‘server_selector’␣==␣" ≪ server_selector ≪
          endl ≪ "‘server_selector␣&␣ZClient::GBV_GVK_ID’␣==␣" ≪
          (server_selector & ZClient::GBV_GVK_ID);
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");      /* START HERE: LDF 2006.10.16. */
#if 0     /* 1 */
      return 0;      /* End of testing. LDF 2006.10.16. */
#endif
      if (perform_search) {
#if 1     /* 0 */
      temp_strm ≪ "Searching.";
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
#endif
      string search_str = search_command;
#if 0     /* 1 */
      temp_strm ≪ "In␣‘zoomtst2’:␣␣‘search_str’␣==␣" ≪ search_str;
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
#endif
      z.search_pqf(const_cast⟨char *⟩(search_str.c_str( )));
```

371. 1108 Elektronische Ressource UND 5550 Vorlesung UND 5550 Video [LDF 2006.07.13.]

```
⟨ Define ZOOM functions 368 ⟩ +≡
  long rec_ctr = 0;
  const char *errmsg;
  const char *addinfo;
  const char *diagset; int error ;
```

372. Block here: only one connection.

```
⟨ Define ZOOM functions 368 ⟩ +≡
  while (ZOOM_event(1, z.get_last_connection( ))) ;
```

373. See if any error occurred.

```
⟨ Define ZOOM functions 368 ⟩ +≡
  if ( ( error = ZOOM_connection_error_x(*(z.get_last_connection( )), &errmsg, &addinfo, &diagset) ) ) {
      temp_strm << "Error:␣" << diagset << ":␣" << errmsg << "(" << error << ")␣" << addinfo;
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str(""); }    /* if */
```

374. OK. Print hit count.

```
⟨ Define ZOOM functions 368 ⟩ +≡
  else {
    rec_ctr = static_cast⟨long⟩(z.get_last_resultset_size( ));
#if 1    /* 0 */
    temp_strm << "Result␣count:␣" << rec_ctr;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
  }    /* else */
  if (write_records_file) {
#if 1    /* 0 */
  temp_strm << "Will␣write␣'records.txt'.";
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  int pos;
  int len;
  int extended_chars;
  const char *rec;
  ofstream out_file;
  out_file.open("records.txt");
```

375. Go through all records at target.

```
⟨ Define ZOOM functions 368 ⟩ +≡
  for (pos = 0; pos < rec_ctr; pos++) { rec =
      ZOOM_record_get(ZOOM_resultset_record(*(z.get_last_resultset( )), pos), "render", &len);
    /* "raw" or "render" syntax */
```

376. If *rec* is non-null, we got a record for display.

```
⟨ Define ZOOM functions 368 ⟩ +≡
  if (rec) {
    temp_strm ≪ pos + 1 ≪ endl;
    if (rec) {
      extended_chars = GetExtendedChars(rec, len);
#if 0   /* 1 */
      temp_strm ≪ rec ≪ "\nlen␣==␣" ≪ len ≪ endl ≪ "extended␣chars␣==␣" ≪ extended_chars;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      out_file ≪ rec ≪ "\n\n";
    }   /* if (rec) */
    else {
#if 0
      temp_strm ≪ rec ≪ "\nlen␣==␣" ≪ len ≪ endl ≪ "extended␣chars␣==␣" ≪ extended_chars;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
    }   /* else */
    temp_strm.str("");
  }   /* if (rec) */
  }   /* for */   /* ***** (5) */
  out_file.close(); }   /* if (write_records_file) */
  else   /* ¬write_records_file */
  {
#if 1   /* 0 */
    temp_strm ≪ "Not␣writing␣'records.txt'.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
  }   /* else (¬write_records_file) */
  }   /* if (perform_search) */
  else   /* (¬perform_search) */
  {
#if 1   /* 0 */
    temp_strm ≪ "Not␣searching.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
  }   /* else (¬perform_search) */
  if (parse_records) {
#if 0   /* 1 */
    temp_strm ≪ "About␣to␣parse␣\"records.txt\".";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    if (clear_database) {
#if 1   /* 0 */
      temp_strm ≪ "Clearing␣database.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
```

```
#endif
        z.clear_database( );
    }     /* if (clear_database) */
    else     /* ¬clear_database */
    {
#if 1     /* 0 */
        temp_strm ≪ "Not␣clearing␣database.";
        AfxMessageBox(temp_strm.str( ).c_str( ));
        temp_strm.str("");
#endif
    }     /* else (¬clear_database) */
    z.parse_records("records.txt");
#if 1     /* 0 */
    temp_strm ≪ "Finished␣parsing␣‘records.txt’.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
  }     /* if (parse_records) */
  else     /* ¬parse_records */
  {
#if 1     /* 0 */
    temp_strm ≪ "Not␣clearing␣database.␣␣Not␣parsing␣‘records.txt’.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
  }     /* else (¬parse_records ) */
```

377.

Log

[LDF 2006.09.06.] Now passing **CString** *curr_search_command_str* to **DB_Display** :: *display_records*.

```
⟨ Define ZOOM functions 368 ⟩ +≡
  if (display_records) { DB_Display db_display;
  db_display.open_html_file("records.html");
  CString curr_search_command_str;
  if (¬perform_search) curr_search_command_str = "No␣search␣performed.";
  else if (¬write_records_file) {
    temp_strm ≪ "Search␣was␣performed,␣but␣records␣file␣wasn't␣written!\n<br>\n" ≪
        "Search␣command:␣␣<tt>" ≪ search_command ≪ "</tt>\n\n";
    curr_search_command_str = temp_strm.str().c_str();
    temp_strm.str("");
  }
  else if (¬parse_records) {
    temp_strm ≪ "Search␣was␣performed␣and␣the␣records␣file␣was␣written,␣" ≪
        "but␣the␣records␣weren't␣written␣to␣the␣database!\n<br>\n" ≪
        "Search␣command:␣␣<tt>" ≪ search_command ≪ "</tt>\n\n";
    curr_search_command_str = temp_strm.str().c_str();
    temp_strm.str("");
  }
  else if (¬clear_database) {
    temp_strm ≪ "Search␣was␣performed,␣records␣file␣was␣written,␣" ≪
        "and␣the␣records␣were␣written␣to␣the␣database,␣but␣the␣" ≪
        "database␣wasn't␣cleared␣before␣writing␣the␣records!\n<br>\n" ≪
        "Search␣command:␣␣<tt>" ≪ search_command ≪ "</tt>\n\n";
    curr_search_command_str = temp_strm.str().c_str();
    temp_strm.str("");
  }
  else {
    temp_strm ≪ "Search␣command:␣␣<tt>" ≪ search_command ≪ "</tt>\n\n";
    curr_search_command_str = temp_strm.str().c_str();
    temp_strm.str("");
  }
#if 0     /* 1 */
  ( ∧ parse_records ∧ write_records_file ∧ clear_database ) ? search_command : "";
#endif
#if 1     /* 0 */
  db_display.display_records(curr_search_command_str);
#else
  db_display.display_records(curr_search_command_str, 26, 27);
#endif
  db_display.close_html_file();
#if 1     /* 0 */
  temp_strm ≪ "Finished␣displaying␣records.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  }     /* if (display_records) */
  else    /* ¬display_records */
  {
#if 1    /* 0 */
    temp_strm ≪ "Not␣displaying␣records.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
  }    /* else (¬display_records) */
```

378. Only do this if I've actually searched for records. [LDF Undated.]

```
⟨Define ZOOM functions 368⟩ +≡
  if (perform_search) {    /* !! TODO: Put code for this in ~ZClient. LDF Undated. */
    ZOOM_resultset_destroy(*(z.get_last_resultset()));
    ZOOM_connection_destroy(*(z.get_last_connection()));
  }    /* if (perform_search) */
  return 0;
#if 0    /* 1 */
  exit(0);
#endif
  }    /* End of zoomtst2 definition. */
```

379. zoomtst3. [LDF Undated.]

```
⟨Declare ZOOM functions 367⟩ +≡
  int zoomtst3(void);
```

380.

```
⟨Define ZOOM functions 368⟩ +≡
  int zoomtst3(void)
  {
    return 0;
  }    /* End of zoomtst3 definition. */
```

381. Putting ZOOM functions together. [LDF 2006.10.05.]

382. This is what's compiled.

```
  ⟨Include files 13⟩
  ⟨ztstzoom.web 364⟩
  ⟨using declarations for namespaces 191⟩
  ⟨Define ZOOM functions 368⟩
```

383. This is what's written to `ztstzoom.h`

⟨ `ztstzoom.h` 383 ⟩ ≡
  ⟨ **using** declarations for namespaces 191 ⟩
  ⟨ Declare ZOOM functions 367 ⟩

384. Output_Stream_Type (`opstrmtp.web`). [LDF 2006.09.18.]

⟨ `opstrmtp.web` 384 ⟩ ≡
  static char *id_string*[] = `"$Id:␣opstrmtp.web,v␣1.5␣2006/10/23␣13:12:32␣Administrator␣Exp␣$"`;

This code is cited in sections 6 and 8.

This code is used in section 394.

385. Preprocessor macro calls. [LDF Undated.]

⟨ Preprocessor macro calls 10 ⟩ +≡
#**pragma** *once*
#**include** `<iostream>`
#**include** `<fstream>`
#**include** `<afxmt.h>`

386. Include files for Output_Stream_Type. [LDF 2006.09.18.]

⟨ Include files 13 ⟩ +≡
#**include** `"stdafx.h"`

387. "Using" declarations for namespaces. [LDF 2006.09.18.]

⟨ "Using" declarations for namespaces 387 ⟩ ≡
  using namespace std;

See also sections 420 and 567.

This code is used in sections 393, 563, and 762.

388. Output_Stream_Type struct declaration. [LDF 2006.09.06.]

Log

[LDF 2006.09.06.] Added this **struct** declaration. It contains the data members **ofstream** *output_stream* and **CMutex** *mutex*, and the functions *lock* and *unlock*.

⟨ Declare **struct Output_Stream_Type** 388 ⟩ ≡
```
struct Output_Stream_Type {
  ofstream output_stream;
  CMutex mutex;
  int lock(void);
  int unlock(void);
};
```
This code is used in section 393.

389. Functions. [LDF 2006.09.06.]

390. Locking. [LDF 2006.09.06.]

Log

[LDF 2006.09.06.] Added this function.

```
⟨ Define Output_Stream_Type functions 390 ⟩ ≡
  int Output_Stream_Type :: lock(void)
  {
    mutex.Lock();
    return 0;
  }
```

See also section 391.

This code is used in section 394.

391. Unlocking. [LDF 2006.09.06.]

Log

[LDF 2006.09.06.] Added this function.

```
⟨ Define Output_Stream_Type functions 390 ⟩ +≡
  int Output_Stream_Type :: unlock(void)
  {
    mutex.Unlock();
    return 0;
  }
```

392. Putting Output_Stream_Type together. [LDF 2006.09.18.]

393. This is what's written to `opstrmtp.h`. [LDF 2006.09.18.]

```
⟨ opstrmtp.h 393 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ "Using" declarations for namespaces 387 ⟩
  ⟨ Declare struct Output_Stream_Type 388 ⟩
```

394. This is what's compiled.

⟨ Include files 13 ⟩
⟨ `opstrmtp.web` 384 ⟩
⟨ Define **Output_Stream_Type** functions 390 ⟩

395. Pica_Record (`picarcrd.web`). [LDF 2006.09.18.]

⟨ `picarcrd.web` 395 ⟩ ≡
  static char *id_string*[] = `"$Id:␣picarcrd.web,v␣1.5␣2006/10/23␣13:13:39␣Administrator␣Exp␣$"`;

This code is cited in sections 6 and 8.

This code is used in section 417.

396. Preprocessor macro calls. [LDF Undated.]

⟨ Preprocessor macro calls 10 ⟩ +≡
#**pragma** *once*

397. Include files for Pica_Record. [LDF 2006.09.18.]

⟨ Include files 13 ⟩ +≡
#**include** `"stdafx.h"`

398. using declarations for namespaces. [LDF 2006.09.18.]

⟨ **using** declarations for namespaces 191 ⟩ +≡
  using namespace std;

399. Forward declarations. [LDF 2006.07.19.]

The declaration of **ZClient** is needed for the **friend** declaration in **class Pica_Record** declaration. [LDF 2006.07.19.]

⟨ Forward declarations 399 ⟩ ≡
  class ZClient;

See also section 768.

This code is used in sections 416 and 776.

400. Pica_Record class declaration. [LDF 2006.09.06.]

Log

[LDF 2006.07.19.] Added this **class** declaration.

[LDF 2006.07.25.] In the declaration of *write_to_database*: Added **ZClient** ∗*zclient* argument.

⟨ Declare **class Pica_Record** 400 ⟩ ≡
  class Pica_Record
  {
  friend ZClient;
protected: **Category_Multimap_Type** *categories*;

See also section 401.

This code is used in section 416.

401.

Log

[LDF 2006.09.06.] In the declaration of *write_to_database*: Changed name of **ofstream** &*log_file* argument to *log_strm*.

[LDF 2006.09.06.] Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

```
⟨ Declare class Pica_Record 400 ⟩ +≡
public: ⟨ Declare Pica_Record functions 403 ⟩
  } ;
```

402. Functions. [LDF 2006.09.06.]

403. Constructor. [LDF Undated.]

```
⟨ Declare Pica_Record functions 403 ⟩ ≡
  Pica_Record(void);
```

See also sections 405, 407, and 413.

This code is used in section 401.

404.

```
⟨ Define Pica_Record functions 404 ⟩ ≡
  Pica_Record::Pica_Record(void)
  {
    return;
  }
```

See also sections 406, 408, 409, 410, 411, 412, and 414.

This code is used in section 417.

405. Destructor. [LDF Undated.]

```
⟨ Declare Pica_Record functions 403 ⟩ +≡
  ~Pica_Record(void);
```

406.

```
⟨ Define Pica_Record functions 404 ⟩ +≡
  Pica_Record :: ~Pica_Record(void)
  {
    return;
  }
```

407. Write to database. [LDF 2006.07.19.]

Log

[LDF 2006.07.19.]	Added this function.
[LDF 2006.07.25.]	Added **ZClient** ∗*zclient* argument.
[LDF 2006.09.06.]	Changed name of **ofstream** &*log_file* argument to *log_strm*.
[LDF 2006.09.06.]	Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

```
⟨ Declare Pica_Record functions 403 ⟩ +≡
  int write_to_database(ZClient *zclient, CDatabase *database, Output_Stream_Type &log_strm);
```

408.

```
⟨ Define Pica_Record functions 404 ⟩ +≡
  int Pica_Record :: write_to_database(ZClient *zclient, CDatabase *database, Output_Stream_Type
          &log_strm){ stringstream temp_strm;
      stringstream sql_strm;
#if 0   /* 1 */
      temp_strm ≪ "Entering␣'Pica_Record::write_to_database'.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      Category_Container *curr_reference_category = 0;
      Subcategory_Container *curr_reference_subcategory = 0;
      if (categories.size() < 1) {
        temp_strm ≪ "ERROR!␣␣In␣'Pica_Record::write_to_database':␣␣" ≪
            "'categories.size()'␣<␣1." ≪ endl ≪ "Exiting␣function␣unsuccessfully␣with␣re\
            turn␣value␣1.";
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
        return 1;
      }   /* if (categories.size() < 1) */
```

409. Get number for the new record. [LDF Undated.]

Log

[LDF 2006.09.06.] !! BUG FIX: Now using *top* 1 in the *select* command contained in the SQL code. Formerly, this could fail, because sorting tables with large

numbers of lines doesn't always work. The problem seems to involve the commmunication of the Visual C++ library functions for ODBC and Microsoft SQL Server 2000.

```
⟨ Define Pica_Record functions 404 ⟩ +≡
  long curr_record_ctr;

  sql_strm ≪ "delete␣Temp_IDs␣" ≪ "insert␣Temp_IDs␣" ≪
      "select␣top␣1␣record_id␣from␣records␣" ≪ "order␣by␣record_id␣desc";
#if 0     /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  database→ExecuteSQL(sql_strm.str( ).c_str( ));
  sql_strm.str("");

  Temp_IDs curr_temp_id;

  curr_temp_id.Open( );
  curr_record_ctr = curr_temp_id.m_temp_id + 1;
#if 0     /* 1 */
  temp_strm ≪ "In␣'Pica_Record::write_to_database':" ≪ endl ≪ "curr_record_ctr␣==␣" ≪
      curr_record_ctr;
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
```

410. Add new record. [LDF Undated.]

```
⟨ Define Pica_Record functions 404 ⟩ +≡
  sql_strm ≪ "set␣identity_insert␣Records␣on␣" ≪ "insert␣Records␣(record_id,␣source_id)␣" ≪
      "values␣(" ≪ curr_record_ctr ≪ ",␣" ≪ zclient→source_id ≪ ")␣" ≪
      "set␣identity_insert␣Records␣off";
  database→ExecuteSQL(sql_strm.str( ).c_str( ));
  sql_strm.str(""); for (Category_Multimap_Type::iterator iter = categories.begin( );
      iter ≠ categories.end( ); ++iter) {
  if (iter→second→subcategory_vector.size( ) < 1) {
    temp_strm ≪ "WARNING!␣␣In␣‘Pica_Record::write_to_database’:␣␣" ≪
        endl ≪ "For␣category␣" ≪ iter→second→pica_plus_category_id ≪
        "␣(" ≪ iter→second→content_description_german ≪ "):" ≪ endl ≪
        "‘subcategory_vectorsize()’␣<␣1." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    continue;
  }   /* if (categories.size( ) < 1) */
  for (Subcategory_Vector_Type::iterator sub_iter = iter→second→subcategory_vector.begin( );
      sub_iter ≠ iter→second→subcategory_vector.end( ); ++sub_iter) { temp_strm ≪ "Category␣==␣" ≪
      iter→second→pica_plus_category_id ≪ "␣" ≪ iter→second→content_description_german ≪
      endl ≪ "␣␣␣␣␣␣pica_plus_field_id␣==␣" ≪ sub_iter→second→pica_plus_field_id ≪
      "␣␣␣␣␣␣field_value␣==␣" ≪ sub_iter→second→field_value ≪ endl;
#if 0
  AfxMessageBox(temp_strm.str( ).c_str( ));
#endif
  temp_strm.str("");
  curr_reference_category = 0;
  curr_reference_subcategory = 0;
  curr_reference_category = zclient→category_map[iter→second→pica_plus_category_id];
      if (curr_reference_category ≠ 0) { curr_reference_subcategory =
      curr_reference_category→subcategory_map[sub_iter→second→pica_plus_field_id];
      if (curr_reference_subcategory ≠ 0) { temp_strm ≪ "curr_reference_cate\
      gory->pica_plus_category_id␣==␣" ≪ curr_reference_category→pica_plus_category_id ≪
      "␣" ≪ curr_reference_category→content_description_german ≪ endl ≪ "curr_reference_subc\
      ategory->pica_plus_field_id␣==␣" ≪ curr_reference_subcategory→pica_plus_field_id ≪
      "␣" ≪ curr_reference_subcategory→content_description_german ≪ endl ≪
      "curr_reference_subcategory->database_commands.size()␣==␣" ≪ static_cast⟨unsigned
      int⟩(curr_reference_subcategory→database_commands.size( ));
#if 0
  AfxMessageBox(temp_strm.str( ).c_str( ));
#endif
  temp_strm.str("");
```

411.

Log

[LDF 2006.09.06.] Now passing **ofstream** &*log_strm* to the **Subcategory_Container** member function referenced by *function*.

[LDF 2006.09.06.] Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

```
⟨ Define Pica_Record functions 404 ⟩ +≡
  for (vector⟨Database_Command ∗⟩::iterator dbc_iter =
          curr_reference_subcategory→database_commands.begin( ); dbc_iter ≠
          curr_reference_subcategory→database_commands.end( ); ++dbc_iter) {
    (∗∗dbc_iter).function(database, curr_record_ctr, iter→second, sub_iter→second, log_strm);
  }     /∗ for (Database_Commands) ∗/
  }     /∗ if (curr_reference_subcategory ≠ 0) ∗/
  }     /∗ if (curr_reference_category ≠ 0) ∗/
  }     /∗ Inner for (Sub-Categories) ∗/
```

412.

Log

[LDF 2006.09.06.] Now passing **ofstream** &*log_strm* to the **Category_Container** member function referenced by *function*.

[LDF 2006.09.06.] Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

```
⟨ Define Pica_Record functions 404 ⟩ +≡
  for (vector⟨Database_Command ∗⟩::iterator dbc_iter =
          curr_reference_category→database_commands.begin( ); dbc_iter ≠
          curr_reference_category→database_commands.end( ); ++dbc_iter) {
    (∗∗dbc_iter).function(database, curr_record_ctr, iter→second, 0, log_strm);
  }     /∗ for (Database_Commands) ∗/
  }     /∗ Outer for (Categories) ∗/
#if 0     /∗ 1 ∗/
  temp_strm ≪ "Exiting␣‘Pica_Record::write_to_database’.";
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  return 0; }     /∗ End of Pica_Record::write_to_database definition. ∗/
```

413. Show. [LDF Undated.]

```
⟨ Declare Pica_Record functions 403 ⟩ +≡
  int show(stringstream &, string = "");
```

414.

```
⟨ Define Pica_Record functions 404 ⟩ +≡
  int Pica_Record::show(stringstream &local_strm, string s)
  {
    if (s ≡ "") s = "Pica_Record:";
    local_strm ≪ s ≪ endl;
    for (Category_Multimap_Type::iterator iter = categories.begin(); iter ≠ categories.end();
          ++iter) {
      iter→second→show(local_strm);
      local_strm ≪ endl;
    }
    return 0;
  }   /* End of Pica_Record::show definition. */
```

415. Putting Pica_Record together. [LDF 2006.09.18.]

416. This is what's written to `picarcrd.h`. [LDF 2006.09.18.]

```
⟨ picarcrd.h 416 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Forward declarations 399 ⟩
  ⟨ Declare class Pica_Record 400 ⟩
```

417. This is what's compiled.

⟨ Include files 13 ⟩
⟨ `picarcrd.web` 395 ⟩
⟨ Define **Pica_Record** functions 404 ⟩

418. Category_Container (`ctgcntnr.web`). [LDF 2006.09.19.]

Log

[LDF 2006.07.19.] Created the file `Category_Container.h`.

[LDF 2006.07.20.] Removed the code for **Subcategory_Container** from the file `Category_Container.h`) and put it into `Subcategory_Container.h`.

[LDF 2006.09.19.] Created this file (`ctgcntnr.web`). It contains the code from `Category_Container.h` and `Category_Container.cpp`, and replaces these files.

```
⟨ ctgcntnr.web 418 ⟩ ≡
  static char id_string[] = "$Id:␣ctgcntnr.web,v␣1.6␣2006/11/28␣16:25:11␣Administrator␣Exp␣$";
```

This code is cited in sections 6 and 8.

This code is used in section 564.

419. Preprocessor macro calls. [LDF Undated.]

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

420. "Using" declarations for namespaces. [LDF 2006.09.19.]

```
⟨ "Using" declarations for namespaces 387 ⟩ +≡
  using namespace std;
```

421. Include files. [LDF Undated.]

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

422. Static variable declarations. [LDF 2006.09.19.]

To Do

Try to eliminate the remaining **static** variables in the interests of thread-safety. [LDF 2006.09.19.]

Log

[LDF 2006.08.16.] Added the **static Temp_IDs** *temp_ids* declaration and removed the declarations of **Temp_IDs** *temp_ids* in the function definitions below.

[LDF 2006.08.22.] Added the declaration of *subject_id*. It's used in `F_041A` and *sub_F_041A*. It can't be declared in `F_041A`, because it needs to be 0 the first time that function is called. In subsequent calls, it shouldn't be 0. It doesn't work to initialize it to 0 in `F_041A`.

[LDF 2006.08.22.] Added the declaration of *subject_id_start*. It's used in `F_041A` and *sub_F_041A*.

[LDF 2006.08.22.] Added the declaration of *previous_repeat_code*. It's used in `F_041A` and *sub_F_041A*.

[LDF 2006.09.19.] Removed the declarations of **stringstream** *temp_strm*, and **stringstream** *sql_strm*, and **Temp_IDs** *temp_ids* as **static** variables. Now declaring them as non-static automatic variables in each function where they're used. I've done this in the interests of thread-safety.

⟨ Declare **static** variables 422 ⟩ ≡
  static long ∗*subject_id* = 0;
  static long ∗*subject_id_start* = 0;
  static unsigned short *previous_repeat_code_ctr* = 0;

This code is used in section 564.

423. Category_Container class declaration. [LDF Undated.]

Log

[LDF 2006.07.19.] Added this **class** declaration.

[LDF 2006.07.19.] Added declaration of assignment operator (**operator**=(**const Category_Container** &*c*)).

[LDF 2006.07.20.] Changed **Subcategory_Map_Type** *subcategories* to
Subcategory_Multimap_Type *subcategories*.

[LDF 2006.07.26.] Added declaration of the function `F_028C`.

[LDF 2006.07.27.] Added declarations of the functions *titles_category_func* and *sub_titles_category_func*.

[LDF 2006.07.27.] Changed the names of *personal_names* and *sub_personal_names* to
personal_names_category_func and *sub_personal_names_category_func*, respectively.

[LDF 2006.07.31.] Made the data members **protected** rather than **public**. This made it necessary to add the **friend** declarations for the classes **Pica_Record**, **ZClient**, and **Subcategory_Container**.

[LDF 2006.08.03.] Added the declaration of `F_001B`.

[LDF 2006.08.15.] Added the declaration of `F_209R`.

[LDF 2006.08.16.] Added the declaration of *sub_F_209R*.

[LDF 2006.08.17.] Added the declaration of `F_209A`.

[LDF 2006.08.18.] Added the declaration of *sub_F_209A*.

[LDF 2006.08.21.] Added the declaration of `F_041A`.

[LDF 2006.08.22.] Added the declaration of *sub_F_041A*.

[LDF 2006.08.22.] Added the declaration of *subject_id*. It's used in `F_041A` and *sub_F_041A*. It can't be declared in `F_041A`, because it needs to be 0 the first time that function is called. In subsequent calls, it shouldn't be 0. It doesn't work to initialize it to 0 in `F_041A`.

[LDF 2006.08.22.] Added the declaration of *subject_id_start*. It's used in `F_041A` and *sub_F_041A*.

[LDF 2006.08.22.] Added the declaration of *previous_repeat_code*. It's used in `F_041A` and *sub_F_041A*.

[LDF 2006.09.06.] Added the **ofstream** &*log_strm* argument to all of the **static** functions that handle **Category_Container** objects.

[LDF 2006.09.06.] Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

[LDF 2006.09.07.] Added the declaration of `F_028B`.

⟨ Declare **class Category_Container** 423 ⟩ ≡
  class Category_Container {

```
    friend class Pica_Record;
    friend class ZClient;
    friend class Subcategory_Container;
  protected: string pica_plus_category_id;
    string pica_3_category_id;
    string content_description_english;
    string content_description_german;
    string repeat_code;
    Subcategory_Map_Type subcategory_map;
    Subcategory_Vector_Type subcategory_vector;
    vector⟨Database_Command *⟩ database_commands;
    vector⟨pair⟨char, string⟩⟩ database_command_arguments;
  public: ⟨ Declare Category_Container functions 426 ⟩
  };
```

This code is used in section 563.

424. Type definitions (typedefs). [2006.07.19.]

Log

[LDF 2006.07.19.] Added this section with the **typedef** for **Category_Map_Type**.

[LDF 2006.07.20.] Added the **typedef** for **Category_Multimap_Type**.

```
  format Category_Map_Type Category_Container
  format Category_Multimap_Type Category_Map_Type
⟨ Type definitions 424 ⟩ ≡
  typedef map⟨string, Category_Container *⟩ Category_Map_Type;
  typedef multimap⟨string, Category_Container *⟩ Category_Multimap_Type;
```

See also section 570.

This code is used in sections 563 and 762.

425. Functions. [LDF 2006.09.19.]

426. Assignment operator. [LDF 2006.09.19.]

Log

[LDF 2006.07.19.] Added this function.

```
⟨ Declare Category_Container functions 426 ⟩ ≡
  void operator=(const Category_Container &c);
```

See also sections 429, 431, 433, 435, 437, 439, 453, 461, 489, 499, 506, 518, 527, 534, 541, 545, 555, and 560.

This code is used in section 423.

427.

⟨ Define **Category_Container** functions 427 ⟩ ≡
```
void Category_Container::operator=(const Category_Container &c)
{
    pica_plus_category_id = c.pica_plus_category_id;
    pica_3_category_id = c.pica_3_category_id;
    content_description_english = c.content_description_english;
    content_description_german = c.content_description_german;
}
```
See also sections 430, 432, 434, 436, 438, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 454, 455, 456, 457, 458, 459, 460, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 490, 491, 492, 493, 494, 495, 496, 497, 498, 500, 501, 502, 503, 504, 505, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 519, 520, 521, 522, 523, 524, 525, 528, 529, 530, 531, 532, 533, 535, 536, 537, 538, 539, 540, 542, 543, 544, 546, 547, 548, 549, 550, 551, 552, 553, 554, 556, 557, 558, 559, and 561.

This code is used in section 564.

428. PICA Category Functions. [LDF 2006.09.19.]

Log

[LDF 2006.09.06.] Added the **ofstream** &*log_strm* argument to all of the **static** functions that handle **Category_Container** objects.

[LDF 2006.09.06.] Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

429. Identifier and Date of the Most Recent Change (F_001B). [LDF 2006.08.03.]

001B **Category_Container**::`F_001B`

Pica+ 001B, Pica3 0210

Kennung und Datum der letzten Änderung

Identifier and Date of the Most Recent Change.

[LDF 2006.08.03.]

There should be exactly two fields in a line with this category, and they should contain three pieces of information: The External Library Number and the date of the most recent change in the `'0'` field, and the time of the most recent change in the `'t'` field. This function performs no error checking, so the information must be present and in the proper format, or it will fail. [LDF 2006.08.03.]

ELN stands for "External Library Number". [LDF 2006.08.17.]

Log

[LDF 2006.08.03.] Added this function.

⟨ Declare **Category_Container** functions 426 ⟩ +≡
```
static int F_001B(
CDatabase *database,
long record_id,
Category_Container *category,
Subcategory_Container *subcategory,
Output_Stream_Type &log_strm);
```

430.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::F_001B(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    stringstream sql_strm;
#if 0     /* 1 */
    temp_strm ≪ "Entering␣‘Category_Container::F_001B’.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    temp_strm ≪ "In␣‘Category_Container::F_001B’." ≪ endl ≪ "Arguments:\n";
    string eln;
    string date;
    string time;
    string::size_type pos;
    for (vector⟨pair⟨char, string⟩⟩::iterator iter = category→database_command_arguments.begin( );
          iter ≠ category→database_command_arguments.end( ); ++iter) {
      temp_strm ≪ iter→first ≪ ":␣" ≪ iter→second ≪ endl;
      if (iter→first ≡ '0') {
        eln = iter→second.substr(0, 4);
        date = iter→second.substr(5, 8);
      }
      else if (iter→first ≡ 't') {
        time = iter→second;
        pos = time.find_last_of('.');
        if (pos ≠ string::npos) time.replace(pos, 1, 1, ':');
      }
      else {
        temp_strm ≪ "WARNING!␣␣In␣‘Category_Container::F_001B’:\n" ≪
            "Invalid␣field.␣␣Will␣try␣to␣continue.";
        AfxMessageBox(temp_strm.str( ).c_str( ));
        temp_strm.str("");
      }
    }   /* for */
    temp_strm ≪ "eln␣==␣" ≪ eln ≪ endl ≪ "date␣==␣" ≪ date ≪ endl ≪ "time␣==␣" ≪ time;
#if 0
    AfxMessageBox(temp_strm.str( ).c_str( ));
#endif
    temp_strm.str("");
    sql_strm ≪ "update␣Records␣" ≪ "set␣date_most_recent_change␣=␣" ≪
        "convert(datetime,␣'" ≪ date ≪ "',␣5)␣" ≪ "+␣convert(datetime,␣'" ≪ time ≪
        "',␣14),␣" ≪ "eln_most_recent_change␣=␣" ≪ eln ≪ "␣where␣record_id␣=␣" ≪ record_id;
    temp_strm ≪ "SQL␣code:␣\n" ≪ sql_strm.str( );
#if 0
    AfxMessageBox(temp_strm.str( ).c_str( ));
#endif
    temp_strm.str("");
    try {
      database→ExecuteSQL(sql_strm.str().c_str());
    }
    catch(CDBException *e)
    {
      temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
          e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
          endl ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
      e→Delete();
    }    /* catch */
    sql_strm.str("");
#if 0    /* 1 */
    temp_strm ≪ "Exiting␣‘Category_Container::F_001B’.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    return 0;
  }    /* End of Category_Container::F_001B definition. */
```

431. Main Canonical Title, Additional Information, Authorship (F_021A). [LDF 2006.09.19.]
021A **Category_Container**::F_021A
Pica+ 021A, Pica3 4000
Hauptsachtitel, Zusätze, Verfasserangabe
Main Canonical Title, Additional Information, Authorship
Pica3 4000

Log

[LDF 2006.07.27.] Added this function.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static int F_021A(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

432.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::F_021A(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    temp_strm ≪ "In␣'Category_Container::F_021A'.";
#if 0
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    titles_category_func("Main_Titles", "Records_Main_Titles", "main_title_id", database, record_id,
        category, log_strm);
    return 0;
  }
```

433. First Author (F_028A). [LDF 2006.09.19.]
Category_Container::F_028A
Pica+ 028A, Pica3 3000
1. Verfasser
First Author

Log

[LDF 2006.07.27.] Removed code from this function. Now calling **Category_Container**::*personal_names_category_func* instead. Also removed the definition of the function *sub_F_028A*, which is no longer needed.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static int F_028A(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

434.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::F_028A(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    temp_strm << "In␣‘Category_Container::F_028A’.";
#if 0
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    personal_names_category_func("Authors", "Records_Authors", "author_id", database, record_id,
        category, log_strm);
    return 0;
  }
```

435. Personal Name, Author (F_028B). [LDF 2006.09.19.]

Pica+ 028B, Pica3 120, 3000–3009

Pica3 120:	Personal Name (Cataloguing form according to RSWK) (RSWK = Regeln für den Schlagwortkatalog = Rules for the Subject Catalogue)
Pica3 3000–3009:	Author
Pica3 120:	Personenname (Ansetzungsform nach RSWK) (RSWK = Regeln für den Schlagwortkatalog)
Pica3 3000–3009:	Verfasser

Log

[LDF 2006.09.07.] Added this function.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static int F_028B(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

436.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::F_028B(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm << "In␣‘Category_Container::F_028B’.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    personal_names_category_func("Authors", "Records_Authors", "author_id", database, record_id,
        category, log_strm);
    return 0;
  }
```

437. Other Contributing Persons (F_028C). [LDF 2006.09.19.]
Category_Container::F_028C
Pica+ 028C, Pica3 301x/302x
Sonstige Beteiligte Personen
Other Contributing Persons
[LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this function.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static int F_028C(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

438.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::F_028C(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0
    temp_strm << "In␣‘Category_Container::F_028C’.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    personal_names_category_func("Contributors", "Records_Contributors", "contributor_id",
        database, record_id, category, log_strm);
    return 0;
  }
```

439. Place, Publisher (F_033A). [LDF 2006.09.19.]
Category_Container::F_033A
Pica+ 033A, Pica3 4030
Ort, Verlag
Place, Publisher
[LDF 2006.08.14.]

Log

[LDF 2006.08.14.] Added this function.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static int F_033A(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

440.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::F_033A(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){ stringstream temp_strm;
#if 0    /* 1 */
      temp_strm ≪ "In␣‘Category_Container::F_033A’.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      bool publisher_switch = false;
      bool place_switch = false;
      bool primary_switch = true;
      string publisher_str = "N/A";
      string place_str = "N/A";
      int r;
```

441.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  for (vector⟨pair⟨char, string⟩⟩::iterator iter = category→database_command_arguments.begin();
      iter ≠ category→database_command_arguments.end(); ++iter) {
#if 0    /* 1 */
  temp_strm ≪ "iter->first:␣" ≪ iter→first ≪ "\niter->second:␣" ≪ iter→second ≪ endl;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

442. Repeated argument. Write the information we already have to the database. [LDF 2006.08.14.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if ((iter→first ≡ 'n' ∧ publisher_switch) ∨ (iter→first ≡ 'p' ∧ place_switch)) {
    r = publishers_database_providers_func(database, record_id, publisher_str, place_str, primary_switch,
        true, log_strm);
#if 0    /* 1 */
    temp_strm ≪ "Resetting␣strings␣and␣switches.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    publisher_str = "N/A";
    place_str = "N/A";
    place_switch = false;
    publisher_switch = false;
    primary_switch = true;
  }    /* if ((iter→first ≡ 'n' ∧ publisher_switch) ∨ (iter→first ≡ 'p' ∧ place_switch)) */
```

443. Handle current argument. [LDF 2006.08.14.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (iter→first ≡ 'n') { publisher_str = iter→second;
#if 0     /* 1 */
  temp_strm ≪ "Setting␣publisher_switch␣to␣true." ≪ "‘publisher_str’␣==␣" ≪ publisher_str ≪
      endl;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

444.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (publisher_str[0] ≡ '[' ∧ *(publisher_str.end() − 1) ≡ ']') {
#if 0     /* 1 */
    temp_strm ≪ "Removing␣brackets␣and␣setting␣‘primary_switch’␣to␣‘false’." ≪ endl;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    publisher_str.erase(0, 1);
    publisher_str.erase(publisher_str.end() − 1);
#if 0     /* 1 */
    temp_strm ≪ "‘publisher_str’␣==␣" ≪ publisher_str ≪ endl;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    primary_switch = false;
  }     /* if (publisher_str is surrounded by brackets.) */
```

445.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  publisher_switch = true; }     /* if (iter→first ≡ 'n') */
```

446.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else if (iter→first ≡ 'p') { place_str = iter→second;
#if 0     /* 1 */
  temp_strm ≪ "Setting␣place_switch␣to␣true." ≪ "‘place_str’␣==␣" ≪ place_str ≪ endl;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

447.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (place_str[0] ≡ '[' ∧ *(place_str.end() − 1) ≡ ']') {
#if 0    /* 1 */
    temp_strm ≪ "Removing␣brackets␣and␣setting␣‘primary_switch’␣to␣‘false’." ≪ endl;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    place_str.erase(0, 1);
    place_str.erase(place_str.end() − 1);
#if 0    /* 1 */
    temp_strm ≪ "‘place_str’␣==␣" ≪ place_str ≪ endl;
#endif
    primary_switch = false;
  }    /* if (place_str is surrounded by brackets.) */
```

448.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  place_switch = true;
#if 0    /* 1 */
  temp_strm ≪ "Setting␣place_switch␣to␣true.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  place_switch = true; }    /* else if (iter→first ≡ 'p') */
```

449. Warning: Invalid database command argument. [LDF 2006.08.14.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else {
    temp_strm ≪ "WARNING!␣␣In␣‘Category_Container::F_033A’:\n" ≪
        "Invalid␣database␣command␣argument:␣␣" ≪ iter→first ≪ ":␣" ≪ iter→second ≪ endl ≪
        "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    continue;
  }    /* else */
```

450.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  }    /* for */
```

451.

Log

[LDF 2006.08.28.] !! BUG FIX: Added this call to **Category_Container**::*publishers_database_providers_func*.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  r = publishers_database_providers_func(database, record_id, publisher_str, place_str, primary_switch, true,
      log_strm);
```

452.

⟨ Define **Category_Container** functions 427 ⟩ +≡
```
  return 0; }    /* End of Category_Container::F_033A definition. */
```

453. Main Subject and Subsidiary Subjects (F_041A). (Subject Assignment) [LDF 2006.09.19.]

Pica+ 041A, Pica3 800, 5100–5199

Pica3 800:	Main Subject and Subsidiary Subjects (Subject Assignment)
Pica3 5100–5199	RSWK Chains (RSWK: Regeln für den Schlagwortkatalog = Rules for the Subject Catalogue)
Pica3 800:	Hauptschlagwort und Unterschlagwörter (Schlagwortansetzung)
Pica3 5100–5199	RSWK-Ketten (RSWK: Regeln für den Schlagwortkatalog)

[LDF 2006.09.19.]
"PPN" stands for "PICA Production Number". [LDF 2006.09.19.]

Log

[LDF 2006.08.21.] Added this function.

⟨ Declare **Category_Container** functions 426 ⟩ +≡
```
  static int F_041A(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

454.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::F_041A(CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){ stringstream temp_strm;
#if 0    /* 1 */
      temp_strm ≪ "In␣‘Category_Container::F_041A’." ≪ endl ≪
          "‘category->repeat_code’␣==␣" ≪ category→repeat_code;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      char subject_type_char = '\0';
      string subject = "N/A";
      long id_number_ppn = 0;
      short chain_number;
      short chain_link_number;
      string chain_info = "N/A";
      string permutation_pattern = "";
      bool permutation_switch = false;
      vector⟨char⟩ subcategory_repeat_vector;
      unsigned short repeat_code_ctr = (category→repeat_code ≠ "") ? atoi(category→repeat_code.c_str()) :
          0;
#if 0    /* 1 */
      temp_strm ≪ "‘repeat_code_ctr’␣==␣" ≪ repeat_code_ctr ≪ endl ≪
          "‘previous_repeat_code_ctr’␣==␣" ≪ previous_repeat_code_ctr;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      if (previous_repeat_code_ctr % 10 ≡ 8 ∧ repeat_code_ctr % 10 ≠ 8) {
#if 0    /* 1 */
        temp_strm ≪ "Deleting␣‘subject_id’␣and␣‘subject_id_start’␣" ≪
            "and␣setting␣them␣to␣0.";
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
        delete subject_id;
        delete subject_id_start;
        subject_id = 0;
        subject_id_start = 0;
      }    /* if */
#if 0    /* 1 */
      else {
        temp_strm ≪ "Not␣deleting␣‘subject_id’␣and␣‘subject_id_start’␣" ≪
            "and␣not␣setting␣them␣to␣0.";
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
      }    /* else */
#endif
      chain_number = repeat_code_ctr/10 + 1;
```

```
        chain_link_number = (repeat_code_ctr % 10 < 6) ? repeat_code_ctr % 10 + 1 : 0;
#if 0      /* 1 */
        temp_strm ≪ "‘repeat_code_ctr’␣==␣" ≪ repeat_code_ctr;
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
        for (vector⟨pair⟨char, string⟩⟩ :: iterator iter = category→database_command_arguments.begin();
              iter ≠ category→database_command_arguments.end(); ++iter) {
#if 0      /* 1 */
        temp_strm ≪ iter→first ≪ ":␣" ≪ iter→second ≪ endl;
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
        if (find(subcategory_repeat_vector.begin(), subcategory_repeat_vector.end(),
              iter→first) ≠ subcategory_repeat_vector.end()) {
#if 0      /* 1 */
          temp_strm ≪ "Already␣had␣" ≪ iter→first ≪ "." ≪ endl;
          AfxMessageBox(temp_strm.str().c_str());
          temp_strm.str("");
#endif
          sub_F_041A(database, record_id, category, repeat_code_ctr, subject_type_char, subject,
                id_number_ppn, chain_number, chain_link_number, chain_info, permutation_pattern,
                permutation_switch, log_strm);
          subject_type_char = '\0';
          subject = "N/A";
          id_number_ppn = 0;
          chain_info = "N/A";
          permutation_pattern = "";
          permutation_switch = false;
          subcategory_repeat_vector.clear();
        }      /* if */
        else {
#if 0      /* 1 */
          temp_strm ≪ "Haven't␣had␣'" ≪ iter→first ≪ "'␣yet." ≪ endl;
          AfxMessageBox(temp_strm.str().c_str());
          temp_strm.str("");
#endif
        }      /* else */
        subcategory_repeat_vector.push_back(iter→first);
```

455. '9' Identifikationsnummer (PPN)
Identification Number (PPN)
[LDF 2006.08.22.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (iter→first ≡ '9') {
    id_number_ppn = atol(iter→second.c_str());
  }
```

456. 'a' Zweites und weiteres Permutationsmuster
Second and additional permutation pattern
[LDF 2006.08.22.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'a' ∧ ((repeat_code_ctr − 8) % 10 ≡ 0)) {
#if 0     /* 1 */
      temp_strm ≪ "Field␣'a':␣␣`repeat_code_ctr'␣==␣" ≪ repeat_code_ctr ≪ ".␣␣It␣ends␣in␣8.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      if (permutation_switch) {
        sub_F_041A(database, record_id, category, repeat_code_ctr, subject_type_char, subject,
            id_number_ppn, chain_number, chain_link_number, chain_info, permutation_pattern,
            permutation_switch, log_strm);
        subject_type_char = '\0';
        subject = "N/A";
        id_number_ppn = 0;
        chain_info = "N/A";
        permutation_pattern = "";
        subcategory_repeat_vector.clear();
      }   /* if (permutation_switch) */
      permutation_switch = true;
      permutation_pattern = iter→second;
    }
```

457. 'a' Angaben zur Schlagwortkette
Subject Chain Information
[LDF 2006.08.22.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'a' ∧ ((repeat_code_ctr − 9) % 10 ≡ 0)) {
#if 0     /* 1 */
      temp_strm ≪ "Field␣'a':␣␣`repeat_code_ctr'␣==␣" ≪ repeat_code_ctr ≪ ".␣␣It␣ends␣in␣9.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      chain_info = iter→second;
    }
```

458. 'a' Hauptschlagwort/Schlagwort
Main or Subsidiary Subject/Subject
LDF 2006.08.22.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'a') {
#if 0    /* 1 */
      temp_strm ≪ "Field␣'a':␣␣`repeat_code_ctr'␣==␣" ≪ repeat_code_ctr ≪
          ".␣␣It␣doesn't␣end␣in␣8␣or␣9.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      subject = iter→second;
    }
```

459. 'f' Permutationsmuster
Permutation Pattern
[LDF 2006.08.22.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'f') {
      permutation_switch = true;
      permutation_pattern = iter→second;
    }
```

460. 'S' Indikator/Schlagwortindikator
Indicator/Subject Indicator
[LDF 2006.08.22.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'S') {
      subject_type_char = iter→second[0];
    }
  }    /* for */
  sub_F_041A(database, record_id, category, repeat_code_ctr, subject_type_char, subject, id_number_ppn,
      chain_number, chain_link_number, chain_info, permutation_pattern, permutation_switch, log_strm);
  previous_repeat_code_ctr = repeat_code_ctr;
  return 0; }    /* End of Category_Container::F_041A definition. */
```

461. sub_F_041A. [LDF 2006.09.19.]
This function is called in `F_041A`. [LDF Undated.]

Log

[LDF 2006.08.22.] Added this function.

⟨ Declare **Category_Container** functions 426 ⟩ +≡
```
static 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);
```

462.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::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){ stringstream temp_strm;
      stringstream sql_strm;
      Temp_IDs temp_ids;
#if 0    /* 1 */
      temp_strm ≪ "In␣‘Category_Container::sub_F_041A’." ≪ endl ≪ "‘repeat_code_ctr’␣==␣" ≪
          repeat_code_ctr ≪ endl ≪ "‘record_id’␣==␣" ≪ record_id ≪ endl;
      if (subject_id) temp_strm ≪ "‘*subject_id’␣==␣" ≪ *subject_id ≪ endl;
      else temp_strm ≪ "‘subject_id’␣is␣null." ≪ endl;
      temp_strm ≪ "‘subject_type_char’␣==␣" ≪ subject_type_char ≪ endl ≪
          "‘subject’␣==␣" ≪ subject ≪ endl ≪ "‘id_number_ppn’␣==␣" ≪
          id_number_ppn ≪ endl ≪ "‘chain_number’␣==␣" ≪ chain_number ≪ endl ≪
          "‘chain_link_number’␣==␣" ≪ chain_link_number ≪ endl ≪ "‘chain_info’␣==␣" ≪
          chain_info ≪ endl ≪ "‘permutation_pattern’␣==␣" ≪ permutation_pattern ≪ endl ≪
          "‘permutation_switch’␣==␣" ≪ permutation_switch;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

463. Error handling: Not enough information to do anything. [LDF 2006.08.22.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (id_number_ppn ≡ 0 ∧ subject ≡ "N/A" ∧ subject_type_char ≡ '\0' ∧ permutation_switch ≡
        false ∧ chain_info ≡ "N/A") {
    temp_strm ≪ "ERROR!␣␣In␣‘sub_F_041A’:" ≪ endl ≪
        "‘id_number_ppn’␣==␣0,␣‘subject’␣==␣\"N/A\",␣" ≪
        "‘subject_type_char’␣==␣'\0',␣and␣‘permutation_switch␣==␣false." ≪
        endl ≪ "Not␣enough␣information␣to␣do␣anything." ≪ endl ≪
        "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    return 1;
  }    /* if */
```

464.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  long subject_type_id = 0; if (subject_type_char ≠ '\0') { sql_strm ≪
      "delete␣Temp_IDs␣insert␣Temp_IDs␣" ≪ "select␣subject_type_id␣from␣Subject_Types␣" ≪
      "where␣indicator␣=␣'" ≪ subject_type_char ≪ "'";
#if 0     /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
```

465.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
```

466.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣`cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }     /* catch */
  sql_strm.str("");
  temp_ids.Open( );
```

467.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (temp_ids.IsBOF( ) ∨ temp_ids.m_temp_id ≡ 0) {
    temp_strm ≪ "ERROR!␣␣In␣`sub_F_041A':" ≪ endl ≪
        "Failed␣to␣find␣a␣`subject_type_id'␣in␣`Subject_Types'␣" ≪
        "that␣corresponds␣to␣`subject_type_char'␣==␣`" ≪ subject_type_char ≪
        "'." ≪ endl ≪ "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    temp_ids.Close( );
    return 1;
  }     /* if (temp_ids.IsBOF( ) ∨ temp_ids.m_temp_id ≡ 0) */
```

468.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else {
    subject_type_id = temp_ids.m_temp_id;
    temp_ids.Close();
#if 0    /* 1 */
    temp_strm ≪ "‘subject_type_id’␣=␣" ≪ subject_type_id;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
  }    /* else */
```

469.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  }    /* if (subject_type_char ≠ '\0') */
```

470. Create a new entry in **Subjects**, if we have enough information. We do this even if it means duplicating entries. The reason is that we store the starting *subject_id* and the ending *subject_id* for a chain of subjects in the *subject_id_start* and *subject_id_end* columns of the **Permutation_Patterns** table, so the *subject_ids* need to be consecutive. [LDF 2006.08.22.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (¬(subject_type_char ≡ '\0' ∧ subject ≡ "N/A" ∧ id_number_ppn ≡ 0 ∧ chain_info ≡ "N/A"))
      { sql_strm ≪ "insert␣Subjects␣(subject_type_id,␣subject,␣" ≪
      "id_number_ppn,␣chain_number,␣chain_link_number,␣" ≪ "chain_info)␣values␣(" ≪
      subject_type_id ≪ ",␣'" ≪ subject ≪ "',␣" ≪ id_number_ppn ≪ ",␣" ≪ chain_number ≪ ",␣" ≪
      chain_link_number ≪ ",␣'" ≪ chain_info ≪ "')";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

471.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
```

472.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }    /* catch */
  sql_strm.str("");
```

473. Now get the *subject_id* from the entry we've just created. [LDF 2006.08.22.]

Log

[LDF 2006.09.01.] !! BUG FIX: Now using *top* 1 in the *select* command contained in the SQL code.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣top␣1␣subject_id␣from␣" ≪
      "Subjects␣order␣by␣subject_id␣desc";
#if 0   /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
```

474.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
```

475.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }   /* catch */
  sql_strm.str("");
```

476.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  temp_ids.Open( );
```

477. Error handling: Failed to retrieve *subject_id* from the **Subjects** table. [LDF 2006.08.22.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (temp_ids.IsBOF( ) ∨ temp_ids.m_temp_id ≡ 0) {
    temp_strm ≪ "ERROR!␣␣In␣‘sub_F_041A’:" ≪ endl ≪
        "Failed␣to␣retrieve␣‘subject_id’␣from␣‘Subjects’." ≪ endl ≪
        "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    temp_ids.Close( );
    return 1;
  }   /* if */
```

478. Set **subject_id*. [LDF 2006.08.22.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (subject_id ≡ 0) {
    subject_id = new long;
    subject_id_start = new long;
    *subject_id_start = temp_ids.m_temp_id;
  }
  *subject_id = temp_ids.m_temp_id;
#if 0     /* 1 */
  temp_strm ≪ "‘*subject_id’␣==␣" ≪ *subject_id;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  temp_ids.Close();
```

479. Add entry to **Records_Subjects**. [LDF 2006.08.22.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  sql_strm ≪ "if␣not␣exists␣(select␣*␣from␣Records_Subjects␣where␣" ≪
      "record_id␣=␣" ≪ record_id ≪ "␣and␣subject_id␣=␣" ≪ *subject_id ≪
      ")␣insert␣Records_Subjects␣(record_id,␣" ≪ "subject_id)␣values␣(" ≪ record_id ≪
      ",␣" ≪ *subject_id ≪ ")";
#if 0     /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

480.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
```

481.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }     /* catch */
  sql_strm.str("");
```

482.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  }     /* if */
```

483. Error handling: *subject_id* $\equiv 0 \land$ *permutation_switch* $\equiv$ *true*. Can't handle this case. A permutation must refer to a subject. [LDF 2006.08.22.]

This case should never arise, because this point shouldn't have been reached unless *subject_id* is non-null. [LDF 2006.08.22.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if ((subject_id_start ≡ 0 ∨ subject_id ≡ 0) ∧ permutation_switch) {
    temp_strm ≪ "ERROR!␣␣In␣‘sub_F_041A’:" ≪ endl ≪ "‘subject_id_start’␣\
        and/or␣‘subject_id’␣is␣null␣" ≪ "and␣‘permutation_switch’␣==␣true." ≪ endl ≪
        "Can’t␣handle␣this␣case.␣␣A␣permutation␣must␣" ≪ "refer␣to␣a␣range␣of␣subjects." ≪
        endl ≪ "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    return 1;
  }   /* if ((subject_id_start ≡ 0 ∨ subject_id ≡ 0) ∧ permutation_switch) */
```

484.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (permutation_switch) {
#if 0   /* 1 */
  temp_strm ≪ "‘*subject_id_start’␣==␣" ≪ *subject_id_start ≪ endl ≪ "‘*subject_id’␣==␣" ≪
      *subject_id;
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  sql_strm ≪ "if␣not␣exists␣(select␣*␣from␣Permutation_Patterns␣" ≪ "where␣record_id␣=␣" ≪
      record_id ≪ "␣and␣subject_id_start␣=␣" ≪ *subject_id_start ≪ "␣and␣subject_id_end␣=␣" ≪
      *subject_id ≪ "␣and␣chain_number␣=␣" ≪ chain_number ≪ "␣and␣permutation_pattern␣=␣’" ≪
      permutation_pattern ≪ "’)␣insert␣Permutation_Patterns␣(record_id,␣subject_id_start,␣" ≪
      "subject_id_end,␣chain_number,␣permutation_pattern)␣values␣(" ≪ record_id ≪ ",␣" ≪
      *subject_id_start ≪ ",␣" ≪ *subject_id ≪ ",␣" ≪ chain_number ≪ ",␣’" ≪ permutation_pattern ≪
      "’)";
#if 0   /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
```

485.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
```

486.

⟨ Define **Category_Container** functions 427 ⟩ +≡

```
catch (CDBException *e)
{
  temp_strm << "Exception␣when␣calling␣'cdb->ExecuteSQL'." << endl << "Error␣code␣=␣" <<
      e->m_nRetCode << endl << "Exception␣==␣" << e->m_strError << endl << "SQL␣Code:" << endl <<
      sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
  e->Delete();
}   /* catch */
sql_strm.str("");
```

487.

⟨ Define **Category_Container** functions 427 ⟩ +≡

```
}   /* if (permutation_switch) */
```

488.

⟨ Define **Category_Container** functions 427 ⟩ +≡

```
if (temp_ids.IsOpen()) temp_ids.Close();
return 0; }   /* End of Category_Container::sub_F_041A definition. */
```

489. Call Number (F_209A). [LDF 2006.09.19.]
Category_Container::F_209A
Pica+ 209A, Pica3 7100–7109
Signatur
Call Number
[LDF 2006.08.17.]

Log

[LDF 2006.08.17.] Added this function.

[LDF 2006.08.18.] Added code to this function. Now calling **Category_Container**::*sub_F_209A*.

⟨ Declare **Category_Container** functions 426 ⟩ +≡

```
static int F_209A(
CDatabase *database,
long record_id,
Category_Container *category,
Subcategory_Container *subcategory,
Output_Stream_Type &log_strm);
```

490.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::F_209A(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){ stringstream temp_strm;
#if 0     /* 1 */
      temp_strm ≪ "Entering␣‘Category_Container::F_209A’." ≪ endl ≪
          "‘category->repeat_code’␣==␣" ≪ category→repeat_code;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      vector⟨char⟩ subcategory_repeat_vector;
      string call_number_str;
      string library_number_str;
      string library_department_str;
      string special_location_str;
      long *call_number_id = 0;
      library_number_str = "0";
      library_department_str = call_number_str = special_location_str = "N/A";
      int r; for (vector⟨pair⟨char,
          string⟩⟩::iterator iter = category→database_command_arguments.begin();
          iter ≠ category→database_command_arguments.end(); ++iter) {
#if 0     /* 1 */
      temp_strm ≪ iter→first ≪ ":␣" ≪ iter→second ≪ endl;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

491. Already had this argument. Process the ones we've collected so far. [LDF 2006.08.18.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (find(subcategory_repeat_vector.begin(), subcategory_repeat_vector.end(),
      iter→first) ≠ subcategory_repeat_vector.end()) {
#if 0     /* 1 */
  temp_strm ≪ "Already␣had␣" ≪ iter→first ≪ "." ≪ endl;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  r = sub_F_209A(database, record_id, category, call_number_str, library_number_str, library_department_str,
      special_location_str, call_number_id, log_strm);
```

492. Clear strings and *subcategory_repeat_vector*. [LDF 2006.08.18.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  library_number_str = "0";
  library_department_str = call_number_str = special_location_str = "N/A";
  subcategory_repeat_vector.clear(); }     /* if */
```

493. Now process current argument. [LDF 2006.08.18.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
```

494. Call Number.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (iter→first ≡ 'a') {
    Subcategory_Container::fix_string(iter→second, call_number_str);
  }
```

495. Library Number.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'b') {
      Subcategory_Container::fix_string(iter→second, library_number_str);
    }
```

496. Special Location.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'f') {
      Subcategory_Container::fix_string(iter→second, special_location_str);
    }
```

497. Library Department.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'j') {
      Subcategory_Container::fix_string(iter→second, library_department_str);
    }
```

498. Warning: Invalid argument.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else {
    temp_strm ≪ "WARNING!␣␣In␣‘Category_Container::F_209A’:" ≪ endl ≪
        "Invalid␣argument:␣␣" ≪ iter→first ≪ ":␣" ≪ iter→second ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    continue;
  }    /* else (Invalid argument) */
  subcategory_repeat_vector.push_back(iter→first); }    /* for */
  r = sub_F_209A(database, record_id, category, call_number_str, library_number_str, library_department_str,
      special_location_str, call_number_id, log_strm);
  delete call_number_id;
  call_number_id = 0;
#if 0    /* 1 */
  temp_strm ≪ "Exiting␣‘Category_Container::F_209A’.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  return 0; }    /* End of Category_Container::F_209A definition. */
```

499. sub_F_209A. [LDF 2006.08.18.]

Log

[LDF 2006.08.18.] Added this function.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static 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);
```

500.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::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){ stringstream temp_strm;
      stringstream sql_strm;
      Temp_IDs temp_ids;
      Call_Numbers call_numbers;
      bool b;
      b = (call_number_id ≡ 0) ? true : false;
#if 0      /* 1 */
      temp_strm ≪ "Entering␣'Category_Container::sub_F_209A'.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

501.

Log

[LDF 2006.08.31.] !! BUG FIX: Now using **Call_Numbers** *call_numbers* instead of writing the *call_number_ids* to **Temp_IDs** and reading *temp_id.m_temp_id*. This worked until the number of call numbers got too large. After that, they weren't sorted properly in the **Temp_IDs** table.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  long temp_call_number_id;
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣call_number_id␣" ≪
      "from␣Call_Numbers␣where␣call_number␣=␣'" ≪ call_number_str ≪
      "'␣and␣library_number␣=␣" ≪ library_number_str ≪ "␣and␣library_department␣=␣'" ≪
      library_department_str ≪ "'␣and␣special_location␣=␣'" ≪ special_location_str ≪ "'";
#if 0     /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣`cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }     /* catch */
  sql_strm.str("");
  temp_ids.Open( ); if (temp_ids.IsBOF( )) {
#if 0     /* 1 */
  temp_strm ≪ "No␣corresponding␣entry␣in␣the␣`Call_Numbers'␣table.␣␣Will␣create␣one.";
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
```

502. *call_number_id* is non-null. This means we already know what the highest value of *call_number_id* is in the **Call_Numbers** table. [LDF 2006.08.18.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (call_number_id) {
    ++(*call_number_id);
#if 0     /* 1 */
    temp_strm ≪ "`call_number_id'␣is␣non-null:␣␣" ≪ "`*call_number_id'␣==␣" ≪
        *call_number_id;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
  }     /* if (call_number_id) */
```

503. *call_number_id* is null. This means we must find the highest value of *call_number_id* in the **Call_Numbers** table. [LDF 2006.08.18.]

Log

[LDF 2006.09.01.] !! BUG FIX: Now using *top* 1 in the *select* command contained in the SQL code.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else    /* call_number_id ≡ 0 */
  {
#if 0    /* 1 */
    temp_strm ≪ "‘call_number_id’␣is␣null.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    sql_strm ≪ "select␣top␣1␣*␣from␣Call_Numbers␣order␣by␣call_number_id␣desc";
#if 0    /* 1 */
    temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    call_numbers.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
    sql_strm.str("");
    call_number_id = new long;
    if (call_numbers.IsBOF( ) ∨ call_numbers.m_call_number_id ≡ 0) {
#if 0    /* 1 */
      temp_strm ≪ "‘call_numbers.m_call_number_id’␣is␣at␣BOF␣or␣0.␣␣" ≪
          "Setting␣‘*call_number’␣to␣1";
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
#endif
      *call_number_id = 1;
    }
    else {
      *call_number_id = call_numbers.m_call_number_id + 1;
      if (*call_number_id ≡ 1) {
#if 0    /* 1 */
        temp_strm ≪ "‘call_numbers.m_call_number_id’␣wasn’t␣at␣BOF␣or␣0,␣" ≪
            "but␣is␣now␣1." ≪ endl ≪ "‘call_numbers.m_call_number_id’␣==␣" ≪
            call_numbers.m_call_number_id;
        AfxMessageBox(temp_strm.str( ).c_str( ));
        temp_strm.str("");
#endif
      }    /* if */
    }    /* else */
    call_numbers.Close( );
#if 0    /* 1 */
    temp_strm ≪ "‘*call_number_id’␣=␣" ≪ *call_number_id;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
  }    /* else (call_number_id ≡ 0) */
```

```
#if 0   /* 1 */
  if (*call_number_id ≡ 1) {
    temp_strm ≪ "'*call_number_id'␣==␣1." ≪ endl ≪ "'record_id'␣==␣" ≪ record_id ≪ endl ≪
         "'call_number_str'␣==␣" ≪ call_number_str ≪ endl;
    if (b) temp_strm ≪ "call_number_id␣was␣null.";
    else temp_strm ≪ "call_number_id␣wasn't␣null.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
  }
#endif
  sql_strm ≪ "set␣identity_insert␣Call_Numbers␣on␣" ≪
      "insert␣Call_Numbers␣(call_number_id,␣" ≪ "call_number,␣library_number,␣library_de\
      partment,␣special_location)␣" ≪ "values␣(" ≪ *call_number_id ≪ ",␣'" ≪ call_number_str ≪
      "',␣" ≪ library_number_str ≪ ",␣'" ≪ library_department_str ≪ "',␣'" ≪ special_location_str ≪
      "')␣" ≪ "set␣identity_insert␣Call_Numbers␣off";
#if 0   /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
        endl ≪ sql_strm.str() ≪ endl ≪ "'record_id'␣==␣" ≪ record_id ≪ endl ≪
        "'*call_number_id'␣==␣" ≪ *call_number_id;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }   /* catch */
  sql_strm.str("");
  temp_call_number_id = *call_number_id; }   /* if (temp_ids.IsBOF()) */
```

504.

⟨Define **Category_Container** functions 427⟩ +≡

```
  else {
#if 0   /* 1 */
    temp_strm ≪ "A␣corresponding␣entry␣in␣the␣'Call_Numbers'␣table␣already␣exists." ≪
        endl ≪ "'temp_ids.m_temp_id'␣==␣" ≪ temp_ids.m_temp_id;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    temp_call_number_id = temp_ids.m_temp_id;
  }   /* else */
```

505. Add an entry to **Records_Call_Numbers**, unless one already exists. [LDF 2006.08.18.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  sql_strm ≪ "if␣not␣exists␣(select␣*␣from␣Records_Call_Numbers␣where␣record_id␣=␣" ≪
      record_id ≪ "␣and␣call_number_id␣=␣" ≪ temp_call_number_id ≪
      ")␣insert␣Records_Call_Numbers␣(record_id,␣call_number_id)␣" ≪ "values␣(" ≪
      record_id ≪ ",␣" ≪ temp_call_number_id ≪ ")";
#if 0     /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }     /* catch */
  sql_strm.str("");
  if (temp_ids.IsOpen( )) temp_ids.Close( );
#if 0     /* 1 */
  temp_strm ≪ "Exiting␣‘Category_Container::sub_F_209A’.";
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  return 0; }     /* End of Category_Container::sub_F_209A definition. */
```

506. Local information regarding remote access to electronic resources (F_209R). [LDF 2006.08.15.] ▮

Category_Container::F_209R
Pica+ 209R, Pica3 7133
Lokale Angaben zum Zugriff auf elektronische Ressourcen im
Fernzugriff
Local information regarding remote access to electronic resources
[LDF 2006.08.15.]

Log

[LDF 2006.08.15.] Added this function.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static int F_209R(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

507.

⟨ Define **Category_Container** functions 427 ⟩ +≡

```
  int Category_Container::F_209R(CDatabase *database, long record_id, Category_Container
        *category, Subcategory_Container *subcategory, Output_Stream_Type &log_strm){
        stringstream temp_strm;
#if 0     /* 1 */
      temp_strm << "In␣'Category_Container::F_209R'.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      vector⟨char⟩ subcategory_repeat_vector;
      string format_str;
      string URL_str;
      string URN_str;
      string license_str;
      string internal_str;
      string web_str;
      long *remote_access_id = 0;
      license_str = "";
      format_str = URL_str = URN_str = internal_str = web_str = "N/A";
      int r; for (vector⟨pair⟨char,
          string⟩⟩::iterator iter = category→database_command_arguments.begin();
          iter ≠ category→database_command_arguments.end(); ++iter) {
#if 0     /* 1 */
      temp_strm << iter→first << ":␣" << iter→second << endl;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

508. Already had this argument. Process the ones we've collected so far. [LDF 2006.08.16.]

⟨ Define **Category_Container** functions 427 ⟩ +≡

```
  if (find(subcategory_repeat_vector.begin(), subcategory_repeat_vector.end(),
        iter→first) ≠ subcategory_repeat_vector.end()) {
#if 0     /* 1 */
  temp_strm << "Already␣had␣" << iter→first << "." << endl;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  r = sub_F_209R(database, record_id, category, format_str, URL_str, URN_str, license_str, internal_str,
        web_str, remote_access_id, log_strm);
```

509. Clear strings and *subcategory_repeat_vector*. [LDF 2006.08.16.]

⟨ Define **Category_Container** functions 427 ⟩ +≡

```
  license_str = "";
  format_str = URL_str = URN_str = internal_str = web_str = "N/A";
  subcategory_repeat_vector.clear(); }     /* if */
```

510. Now process current argument. [LDF 2006.08.16.]

⟨ Define **Category_Container** functions 427 ⟩ +≡

511. Format

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (iter→first ≡ 'O') {
    Subcategory_Container::fix_string(iter→second, format_str);
  }
```

512. URL

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'a') {
      Subcategory_Container::fix_string(iter→second, URL_str);
    }
```

513. URN

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'g') {
      Subcategory_Container::fix_string(iter→second, URN_str);
    }
```

514. License indicator

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'S') {
      Subcategory_Container::fix_string(iter→second, license_str);
    }
```

515. Internal Remarks

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'x')      /* Internal Remarks */
    {
      Subcategory_Container::fix_string(iter→second, internal_str);
    }
```

516. Text for the Web Display

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else
    if (iter→first ≡ 'y') {
      Subcategory_Container::fix_string(iter→second, web_str);
    }
```

517. Warning: Invalid argument.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else {
    temp_strm ≪ "WARNING!␣␣In␣‘Category_Container::F_209R’:" ≪ endl ≪
        "Invalid␣argument:␣␣" ≪ iter→first ≪ ":␣" ≪ iter→second ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    continue;
  }    /* else (Invalid argument) */
  subcategory_repeat_vector.push_back(iter→first); }    /* for */
  r = sub_F_209R(database, record_id, category, format_str, URL_str, URN_str, license_str, internal_str,
      web_str, remote_access_id, log_strm);
  delete remote_access_id;
  return 0; }    /* End of Category_Container::F_209R definition. */
```

518. **sub_F_209R.** [LDF 2006.09.19.]

Log

[LDF 2006.08.15.] Added this function.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static 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);
```

519.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::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){ stringstream temp_strm;
      stringstream sql_strm;
      Temp_IDs temp_ids;
#if 0   /* 1 */
      temp_strm ≪ "In␣‘Category_Container::sub_F_209R’.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
#if 0   /* 1 */
      temp_strm ≪ "record_id␣=␣" ≪ record_id ≪ endl ≪ "format_str␣=␣" ≪ format_str ≪
          endl ≪ "URL_str␣==␣" ≪ URL_str ≪ endl ≪ "URN_str␣==␣" ≪ URN_str ≪ endl ≪
          "license_str␣==␣" ≪ license_str ≪ endl ≪ "internal_str␣==␣" ≪ internal_str ≪ endl ≪
          "web_str␣==␣" ≪ web_str;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

520. Error handling: *license_str* ≡ "". [LDF 2006.08.16.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
#if 0   /* 1 */
  if (license_str ≡ "") {
    temp_strm ≪ "ERROR!␣␣In␣‘sub_F_209R’:␣␣‘license_str’␣==␣\"\"" ≪ endl ≪
        "This␣isn’t␣allowed.␣␣Category␣209R␣must␣have␣a␣non-null␣license␣field." ≪ endl ≪
        "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    return 1;
  }   /* if (license_str ≡ "") */
#endif   /* START HERE: LDF 2006.09.04. Fix this. */
  if (license_str ≡ "") license_str = "0";
```

521. Find out whether a corresponding entry in the **Remote_Access** table already exists. [LDF 2006.08.16.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  long temp_remote_access_id;
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣remote_access_id␣from␣" ≪
      "Remote_Access␣where␣license_indicator␣=␣'" ≪ license_str ≪ "'␣and␣format_type␣=␣'" ≪
      format_str ≪ "'␣and␣web_display_text␣=␣'" ≪ web_str ≪ "'␣and␣URL␣=␣'" ≪ URL_str ≪
      "'␣and␣URN␣=␣'" ≪ URN_str ≪ "'␣and␣internal_remarks␣=␣'" ≪ internal_str ≪ "'";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_strm.str("");
  temp_ids.Open( ); if (temp_ids.IsBOF( )) {
#if 0    /* 1 */
  temp_strm ≪ "No␣corresponding␣entry␣in␣the␣'Remote_Access'␣table.␣␣Will␣create␣one.";
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
```

522. *remote_access_id* is non-null. This means we already know what the highest value of *remote_access_id* is in the **Remote_Access** table. [LDF 2006.08.16.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (remote_access_id) {
    ++(*remote_access_id);
#if 0    /* 1 */
    temp_strm ≪ "'remote_access_id'␣is␣non-null:␣␣" ≪ "'*remote_access_id'␣==␣" ≪
        *remote_access_id;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
  }    /* if (remote_access_id) */
```

523. *remote_access_id* is null. This means we must find the highest value of *remote_access_id* in the **Remote_Access** table. [LDF 2006.08.16.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else    /* remote_access_id ≡ 0 */
  {
```

524.

Log

[LDF 2006.09.01.] !! BUG FIX: Now using *top* 1 in the *select* command contained in the SQL code.

```
⟨ Define Category_Container functions 427 ⟩ +≡
#if 0    /* 1 */
  temp_strm ≪ "‘remote_access_id’␣is␣null.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  temp_ids.Close();
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣" ≪ "select␣top␣1␣remote_access_id␣from␣Remo\
      te_Access␣" ≪ "order␣by␣remote_access_id␣desc";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }    /* catch */
  sql_strm.str("");
  temp_ids.Open();
  remote_access_id = new long;
  if (temp_ids.IsBOF() ∨ temp_ids.m_temp_id ≡ 0) *remote_access_id = 1;
  else *remote_access_id = temp_ids.m_temp_id + 1;
#if 0    /* 1 */
  temp_strm ≪ "‘*remote_access_id’␣==␣" ≪ *remote_access_id;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  }    /* else (remote_access_id ≡ 0) */
  sql_strm ≪ "set␣identity_insert␣Remote_Access␣on␣" ≪
      "insert␣Remote_Access␣(remote_access_id,␣" ≪
      "license_indicator,␣format_type,␣web_display_text,␣URL,␣URN,␣" ≪
      "internal_remarks)␣values␣(" ≪ *remote_access_id ≪ ",␣" ≪ license_str ≪ ",␣’" ≪
      format_str ≪ "’,␣’" ≪ web_str ≪ "’,␣’" ≪ URL_str ≪ "’,␣’" ≪ URN_str ≪ "’,␣’" ≪
      internal_str ≪ "’)␣" ≪ "set␣identity_insert␣Remote_Access␣off";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }   /* catch */
  sql_strm.str("");
  temp_remote_access_id = *remote_access_id; }   /* if (temp_ids.IsBOF()) */
  else {
#if 0   /* 1 */
    temp_strm ≪ "A␣corresponding␣entry␣in␣the␣‘Remote_Access’␣table␣already␣exists." ≪
        endl ≪ "‘temp_ids.m_temp_id’␣==␣" ≪ temp_ids.m_temp_id;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    temp_remote_access_id = temp_ids.m_temp_id;
  }   /* else */
```

525. Add an entry to **Records_Remote_Access**, unless one already exists. [LDF 2006.08.16.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  sql_strm ≪ "if␣not␣exists␣(select␣*␣from␣Records_Remote_Access␣where␣record_id␣=␣" ≪
      record_id ≪ "␣and␣remote_access_id␣=␣" ≪ temp_remote_access_id ≪
      ")␣insert␣Records_Remote_Access␣(record_id,␣remote_access_id)␣" ≪
      "values␣(" ≪ record_id ≪ ",␣" ≪ temp_remote_access_id ≪ ")";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_strm.str("");
  if (temp_ids.IsOpen( )) temp_ids.Close( );
  return 0; }    /* End of Category_Container::sub_F_209R definition. */
```

526. Functions for groups of categories. [LDF 2006.09.19.]

527. personal_names_category_func. [LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this function.

[LDF 2006.07.27.] Changed the name of this function from *personal_names* to *personal_names_category_func*.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static int personal_names_category_func(
  string main_table_name,
  string assoc_table_name,
  string column_name,
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Output_Stream_Type &log_strm);
```

528.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::personal_names_category_func(
  string main_table_name,
  string assoc_table_name,
  string column_name,
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Output_Stream_Type &log_strm){
      stringstream temp_strm;
#if 0     /* 1 */
      temp_strm ≪ "In␣‘personal_names_category_func’:␣␣" ≪ "‘main_table_name’␣==␣" ≪
          main_table_name;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      string curr_surname;
      string curr_given_name = "N/A";
      string curr_prefix = "N/A";
      unsigned long curr_id_number_ppn = 0;
      long table_ctr = 0;
      vector⟨char⟩ used_fields;
      vector⟨char⟩::iterator used_fields_iter;
#if 0     /* 1 */
      temp_strm ≪ "In␣‘Category_Container::personal_names_category_func’:␣" ≪ endl ≪
          "‘record_id’␣==␣" ≪ record_id ≪ endl ≪ "Arguments:";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

529. No arguments. This shouldn't happen. If it does, issue a warning and return 1. [LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this section.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (category→database_command_arguments.size() ≤ 0) {
    temp_strm ≪ "WARNING!␣␣In␣‘Category_Container::personal_names_category_func’:" ≪
        endl ≪ "No␣arguments.␣␣This␣shouldn’t␣happen." ≪
        "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    return 1;
  }    /* if (No arguments) */
```

530. Extract the arguments from *database_command_arguments*. [LDF 2006.07.25.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  for (vector⟨pair⟨char, string⟩⟩ :: iterator iter = category→database_command_arguments.begin( );
        iter ≠ category→database_command_arguments.end( ); ++iter) {
#if 0    /* 1 */
  temp_strm ≪ ":␣" ≪ iter→second ≪ endl;
#endif
```

531. Check whether we've already had this argument before. If we have, we need to write database entries using the old arguments before we process the next set, starting with the current argument. [LDF 2006.07.26.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if ((used_fields_iter = find(used_fields.begin( ), used_fields.end( ), iter→first)) ≡ used_fields.end( )) {
#if 0    /* 1 */
    temp_strm ≪ "'" ≪ iter→first ≪ "'␣hasn't␣been␣used␣already." ≪ endl;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    used_fields.push_back(iter→first);
  }    /* if */
```

532.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else    /* Found duplicate field */
  {
#if 0    /* 1 */
    temp_strm ≪ "'" ≪ iter→first ≪ "'␣has␣been␣used␣already." ≪ endl;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    table_ctr = sub_personal_names_category_func(main_table_name, assoc_table_name, column_name,
          database, record_id, category, curr_surname, curr_given_name, curr_prefix, curr_id_number_ppn,
          table_ctr, log_strm);
    used_fields.clear( );
    used_fields.push_back(iter→first);
    curr_surname = "";
    curr_given_name = "N/A";
    curr_prefix = "N/A";
    curr_id_number_ppn = 0;
  }    /* else (Found duplicate field) */
```

533.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (iter→first ≡ '9') {
    curr_id_number_ppn = strtol(iter→second.c_str( ), 0, 10);
#if 0      /* 1 */
    temp_strm ≪ "curr_id_number_ppn␣==␣" ≪ curr_id_number_ppn ≪ endl;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
  }
  else if (iter→first ≡ 'a') curr_surname = iter→second;
  else if (iter→first ≡ 'd') curr_given_name = iter→second;
  else if (iter→first ≡ 'c') curr_prefix = iter→second;
  }    /* for */
  table_ctr = sub_personal_names_category_func(main_table_name, assoc_table_name, column_name,
      database, record_id, category, curr_surname, curr_given_name, curr_prefix, curr_id_number_ppn,
      table_ctr, log_strm);
  return 0; }    /* End of Category_Container::personal_names_category_func definition. */
```

534. sub_personal_names_category_func. [LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this function.

[LDF 2006.07.27.] Changed the name of this function from *sub_personal_names* to *sub_personal_names_category_func*.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static int sub_personal_names_category_func(
  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);
```

535.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container :: sub_personal_names_category_func(
  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)
      {
      stringstream temp_strm;
      stringstream sql_strm;
      Temp_IDs temp_ids;
```

536. Check whether there's already an entry for this person in the main table. [LDF 2006.07.25.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣" ≪ column_name ≪ "␣from␣" ≪
      main_table_name ≪ "␣where␣given_name␣=␣'" ≪ given_name ≪ "'␣and␣" ≪ "surname␣=␣'" ≪
      surname ≪ "'" ≪ "␣and␣id_number_ppn␣=␣" ≪ id_number_ppn;
  if (prefix ≠ "") sql_strm ≪ "␣and␣prefix␣=␣'" ≪ prefix ≪ "'";
#if 0    /* 1 */
  temp_strm ≪ "sql_strm.str()␣==␣\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣`cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_strm.str("");
  temp_ids.Open( );
```

537. No entry for this author in the main table, so we create one. [LDF 2006.07.25.]
Get *table_id*. [LDF 2006.07.26.]

Log

[LDF 2006.09.01.] !! BUG FIX: Now using *top* 1 in the *select* command contained in the SQL code.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  long table_id; if (temp_ids.IsBOF() ∨ temp_ids.m_temp_id ≡ 0) {
  if (table_ctr ≡ 0) {
    temp_ids.Close();
    sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣top␣1␣" ≪ column_name ≪ "␣from␣" ≪
        main_table_name ≪ "␣order␣by␣" ≪ column_name ≪ "␣desc";
#if 0    /* 1 */
    temp_strm ≪ "sql_strm.str()␣==␣\n" ≪ sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    try {
      database→ExecuteSQL(sql_strm.str().c_str());
    }
    catch(CDBException *e)
    {
      temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
          e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
          endl ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
      e→Delete();
    }    /* catch */
    sql_strm.str("");
    temp_ids.Open();
    if (temp_ids.IsBOF()) table_id = 1;
    else table_id = temp_ids.m_temp_id + 1;
#if 0    /* 1 */
    temp_strm ≪ "After␣query:␣␣'temp_ids.m_temp_id'␣==␣" ≪ temp_ids.m_temp_id ≪ endl ≪
        "'table_id'␣==␣" ≪ table_id;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
  }
  else table_id = table_ctr;
```

538. Enter data into main table. [LDF 2006.07.25.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  sql_strm ≪ "set␣identity_insert␣" ≪ main_table_name ≪ "␣on␣" ≪
      "insert␣" ≪ main_table_name ≪ "␣(" ≪ column_name ≪
      ",␣given_name,␣surname,␣prefix,␣id_number_ppn)␣" ≪ "values␣(" ≪ table_id ≪ ",␣'" ≪
      given_name ≪ "',␣'" ≪ surname ≪ "',␣'" ≪ prefix ≪ "',␣" ≪ id_number_ppn ≪ ")␣" ≪
      "set␣identity_insert␣" ≪ main_table_name ≪ "␣off";
#if 0    /* 1 */
  temp_strm ≪ "sql_strm␣==␣" ≪ endl ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_strm.str(""); }    /* if (temp_ids.IsBOF( ) ∨ temp_ids.m_temp_id ≡ 0) */
  else table_id = temp_ids.m_temp_id;
```

539. Check whether an entry for this combination of record and person exists in the association table. If it doesn't, add one. [LDF 2006.07.26.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  temp_ids.Close();
#if 0    /* 1 */
  temp_strm ≪ "About␣to␣refill␣Temp_IDs.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  sql_strm ≪ "delete␣Temp_IDs␣" ≪ "insert␣Temp_IDs␣select␣record_id␣from␣" ≪
      assoc_table_name ≪ "␣where␣record_id␣=␣" ≪ record_id ≪ "␣and␣" ≪ column_name ≪ "␣=␣" ≪
      table_id;
#if 0    /* 1 */
  temp_strm ≪ "sql_strm␣==␣" ≪ endl ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }    /* catch */
  sql_strm.str("");
  temp_ids.Open();
```

540. No corresponding entry exists in the association table, so we create one. [LDF 2006.07.26.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (temp_ids.IsBOF() ∨ temp_ids.m_temp_id ≡ 0) {
    sql_strm ≪ "insert␣" ≪ assoc_table_name ≪ "␣(record_id,␣" ≪ column_name ≪ ")␣" ≪
        "values␣(" ≪ record_id ≪ ",␣" ≪ table_id ≪ ")";
#if 0    /* 1 */
    temp_strm ≪ "Inserting␣entry␣into␣" ≪ assoc_table_name ≪ ":" ≪ endl ≪ "sql_strm␣==␣" ≪
        endl ≪ sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    try {
      database→ExecuteSQL(sql_strm.str().c_str());
    }
    catch(CDBException *e)
    {
      temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
          e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
          endl ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
      e→Delete();
    }    /* catch */
    sql_strm.str("");
  }
  temp_ids.Close();
  return table_id + 1; }
    /* End of Category_Container::sub_personal_names_category_func definition. */
```

541. titles_category_func. [LDF 2006.07.27.]

Log

[LDF 2006.07.27.] Added this function.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static int titles_category_func(
  string main_table_name,
  string assoc_table_name,
  string column_name,
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Output_Stream_Type &log_strm);
```

542.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::titles_category_func(
  string main_table_name,
  string assoc_table_name,
  string column_name,
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Output_Stream_Type &log_strm)
       {
       stringstream temp_strm;
#if 0     /* 1 */
       temp_strm ≪ "In␣‘titles_category_func’:␣␣" ≪ "‘main_table_name’␣==␣" ≪
            main_table_name;
       AfxMessageBox(temp_strm.str().c_str());
       temp_strm.str("");
#endif
       vector⟨char⟩ used_fields;
       vector⟨char⟩::iterator used_fields_iter;
       const static unsigned short MAIN = 1;
       const static unsigned short PARALLEL = 2;
       unsigned short state = MAIN;
#if 0     /* 1 */
       temp_strm ≪ "In␣‘Category_Container::titles_category_func’:␣" ≪ endl ≪
            "‘record_id’␣==␣" ≪ record_id ≪ endl ≪ "‘state’␣==␣" ≪ state ≪ endl ≪
            static_cast⟨unsigned int⟩(category→database_command_arguments.size()) ≪ "␣arguments.";
       AfxMessageBox(temp_strm.str().c_str());
       temp_strm.str("");
#endif
```

543. No arguments. This shouldn't happen. If it does, issue a warning and return 1. [LDF 2006.07.27.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (category→database_command_arguments.size() ≤ 0) {
    temp_strm ≪ "WARNING!␣␣In␣‘Category_Container::titles_category_func’:" ≪
        endl ≪ "No␣arguments.␣␣This␣shouldn’t␣happen." ≪
        "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    return 1;
  }    /* if (No arguments) */
  long table_ctr = 0;
  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; for (vector⟨pair⟨char,
      string⟩⟩::iterator iter = category→database_command_arguments.begin();
      iter ≠ category→database_command_arguments.end(); ++iter) {
#if 0    /* 1 */
  temp_strm ≪ iter→first ≪ ":␣" ≪ iter→second ≪ endl;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

544. Check whether we've already had this argument before. If we have, we need to write database entries using the old arguments before we process the next set, starting with the current argument. [LDF 2006.07.27.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if ((used_fields_iter = find(used_fields.begin(), used_fields.end(), iter→first)) ≡ used_fields.end()) {
#if 0    /* 1 */
    temp_strm ≪ "'" ≪ iter→first ≪ "'␣hasn't␣been␣used␣already." ≪ endl;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    used_fields.push_back(iter→first);
  }    /* if */
  else    /* Found duplicate field */
  {
#if 0    /* 1 */
    temp_strm ≪ "'" ≪ iter→first ≪ "'␣has␣been␣used␣already." ≪ endl;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    table_ctr = sub_titles_category_func(main_table_name, assoc_table_name, column_name, database,
        record_id, category, table_ctr, log_strm, main_canonical_title, continuation_main_canonical_title,
        additions_main, continuation_additions_main, authorship, standard_text, additional_creator_main,
        parallel_canonical_title, additions_parallel, additional_creator_parallel);
    used_fields.clear();
    used_fields.push_back(iter→first);
    delete main_canonical_title;
    delete continuation_main_canonical_title;
    delete additions_main;
    delete continuation_additions_main;
    delete authorship;
    delete standard_text;
    delete additional_creator_main;
    delete parallel_canonical_title;
    delete additions_parallel;
    delete additional_creator_parallel;
    main_canonical_title = 0;
    continuation_main_canonical_title = 0;
    additions_main = 0;
    continuation_additions_main = 0;
    authorship = 0;
    standard_text = 0;
    additional_creator_main = 0;
    parallel_canonical_title = 0;
    additions_parallel = 0;
    additional_creator_parallel = 0;
  }    /* else (Found duplicate field) */
  if (iter→first ≡ '1') {
    standard_text = new string;
    *standard_text = iter→second;
  }
  else if (iter→first ≡ '!') {
    continuation_main_canonical_title = new string;
    *continuation_main_canonical_title = iter→second;
  }
  else if (iter→first ≡ '?') {
    continuation_additions_main = new string;
    *continuation_additions_main = iter→second;
  }
  else if (iter→first ≡ 'a') {
    state = MAIN;
    main_canonical_title = new string;
    *main_canonical_title = iter→second;
  }
  else if (iter→first ≡ 'f') {
    state = PARALLEL;
    parallel_canonical_title = new string;
    *parallel_canonical_title = iter→second;
  }
  else if (iter→first ≡ 'h') {
    authorship = new string;
    *authorship = iter→second;
  }
  else if (iter→first ≡ 'd' ∧ state ≡ MAIN) {
    additions_main = new string;
    *additions_main = iter→second;
  }
  else if (iter→first ≡ 'e' ∧ state ≡ MAIN) {
    additional_creator_main = new string;
    *additional_creator_main = iter→second;
  }
  else if (iter→first ≡ 'd' ∧ state ≡ PARALLEL) {
    additions_parallel = new string;
    *additions_parallel = iter→second;
  }
  else if (iter→first ≡ 'e' ∧ state ≡ PARALLEL) {
    additional_creator_parallel = new string;
    *additional_creator_parallel = iter→second;
  }
  }    /* for */
  table_ctr = sub_titles_category_func(main_table_name, assoc_table_name, column_name, database,
      record_id, category, table_ctr, log_strm, main_canonical_title, continuation_main_canonical_title,
      additions_main, continuation_additions_main, authorship, standard_text, additional_creator_main,
      parallel_canonical_title, additions_parallel, additional_creator_parallel);
  return 0; }     /* End of Category_Container :: titles_category_func definition. */
```

545. sub_titles_category_func. [LDF Undated.]

Log

[LDF 2006.07.27.] Added this function.

[LDF 2006.09.04.] Added the optional **string** *∗continuation* argument with the default 0.

[LDF 2006.09.05.] Changed **string** *∗continuation* to *continuation_main_canonical_title*. Added the optional **string** *∗continuation_additions_main* argument with the default 0.

```
⟨ Declare Category_Container functions 426 ⟩ +≡
  static int sub_titles_category_func(
  string main_table_name,
  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);
```

546.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::sub_titles_category_func(
  string main_table_name,
  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,
  string *continuation_main_canonical_title,
  string *additions_main,
  string *continuation_additions_main,
  string *authorship,
  string *standard_text,
  string *additional_creator_main,
  string *parallel_canonical_title,
  string *additions_parallel,
  string *additional_creator_parallel)
      {
      stringstream temp_strm;
      stringstream sql_strm;
      Temp_IDs temp_ids;
#if 0    /* 1 */
      temp_strm ≪ "In␣'Category_Container::sub_titles_category_func'.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      stringstream column_strm;
      stringstream value_strm;
      long table_id;
```

547. Construct SQL queries. [LDF 2006.07.27.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  string and_str = "";
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣" ≪ column_name ≪ "␣from␣" ≪
      main_table_name ≪ "␣where␣";
  if (main_canonical_title) {
    sql_strm ≪ "main_canonical_title␣=␣'" ≪ *main_canonical_title ≪ "'";
    and_str = "␣and␣";
    column_strm ≪ ",␣main_canonical_title";
    value_strm ≪ ",␣'" ≪ *main_canonical_title ≪ "'";
  }
  if (continuation_main_canonical_title) {
    sql_strm ≪ and_str ≪ "continuation_main_canonical_title␣=␣'" ≪
        *continuation_main_canonical_title ≪ "'";
    and_str = "␣and␣";
    column_strm ≪ ",␣continuation_main_canonical_title";
    value_strm ≪ ",␣'" ≪ *continuation_main_canonical_title ≪ "'";
  }
  if (additions_main) {
    sql_strm ≪ and_str ≪ "additions_main␣=␣'" ≪ *additions_main ≪ "'";
    and_str = "␣and␣";
    column_strm ≪ ",␣additions_main";
    value_strm ≪ ",␣'" ≪ *additions_main ≪ "'";
  }
  if (continuation_additions_main) {
    sql_strm ≪ and_str ≪ "continuation_additions_main␣=␣'" ≪ *continuation_additions_main ≪ "'";
    and_str = "␣and␣";
    column_strm ≪ ",␣continuation_additions_main";
    value_strm ≪ ",␣'" ≪ *continuation_additions_main ≪ "'";
  }
  if (authorship) {
    sql_strm ≪ and_str ≪ "authorship␣=␣'" ≪ *authorship ≪ "'";
    and_str = "␣and␣";
    column_strm ≪ ",␣authorship";
    value_strm ≪ ",␣'" ≪ *authorship ≪ "'";
  }
  if (standard_text) {
    sql_strm ≪ and_str ≪ "standard_text␣=␣'" ≪ *standard_text ≪ "'";
    and_str = "␣and␣";
    column_strm ≪ ",␣standard_text";
    value_strm ≪ ",␣'" ≪ *standard_text ≪ "'";
  }
  if (additional_creator_main) {
    sql_strm ≪ and_str ≪ "additional_creator_main␣=␣'" ≪ *additional_creator_main ≪ "'";
    and_str = "␣and␣";
    column_strm ≪ ",␣additional_creator_main";
    value_strm ≪ ",␣'" ≪ *additional_creator_main ≪ "'";
  }
  if (parallel_canonical_title) {
    sql_strm ≪ and_str ≪ "parallel_canonical_title␣=␣'" ≪ *parallel_canonical_title ≪ "'";
    and_str = "␣and␣";
    column_strm ≪ ",␣parallel_canonical_title";
    value_strm ≪ ",␣'" ≪ *parallel_canonical_title ≪ "'";
  }
  if (additions_parallel) {
    sql_strm ≪ and_str ≪ "additions_parallel␣=␣'" ≪ *additions_parallel ≪ "'";
    and_str = "␣and␣";
    column_strm ≪ ",␣additions_parallel";
    value_strm ≪ ",␣'" ≪ *additions_parallel ≪ "'";
  }
  if (additional_creator_parallel) {
    sql_strm ≪ and_str ≪ "additional_creator_parallel␣=␣'" ≪ *additional_creator_parallel ≪ "'";
    and_str = "␣and␣";
    column_strm ≪ ",␣additional_creator_parallel";
    value_strm ≪ ",␣'" ≪ *additional_creator_parallel ≪ "'";
  }
#if 0    /* 1 */
  temp_strm ≪ "sql_strm.str()␣==␣\n" ≪ sql_strm.str() ≪ endl ≪ "column_strm.str()␣==␣\n" ≪
      column_strm.str() ≪ endl ≪ "value_strm.str()␣==␣\n" ≪ value_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

548. Find out whether there's already an identical entry in **Main_Titles**. [LDF 2006.07.27.]

```
⟨Define Category_Container functions 427⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }    /* catch */
  sql_strm.str("");
  temp_ids.Open();
```

549. Create an entry in **Main_Titles**. [LDF 2006.07.27.]

```
⟨Define Category_Container functions 427⟩ +≡
  if (temp_ids.IsBOF() ∨ temp_ids.m_temp_id ≡ 0) {
#if 0    /* 1 */
  temp_strm ≪ "Creating␣a␣new␣entry␣in␣'Main_Titles'.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

550.

Log

[LDF 2006.09.01.] !! BUG FIX: Now using *top* 1 in the *select* command contained in the SQL code.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (table_ctr ≡ 0) {
    temp_ids.Close();
    sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣top␣1␣" ≪ column_name ≪ "␣from␣" ≪
        main_table_name ≪ "␣order␣by␣" ≪ column_name ≪ "␣␣desc";
#if 0    /* 1 */
    temp_strm ≪ "About␣to␣call␣SQL␣command:\n" ≪ sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    try {
      database→ExecuteSQL(sql_strm.str().c_str());
    }
    catch(CDBException *e)
    {
      temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
          e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
          endl ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
      e→Delete();
    }    /* catch */
    sql_strm.str("");
    temp_ids.Open();
    if (temp_ids.IsBOF()) {
      table_id = 1;
    }
    else {
      table_id = temp_ids.m_temp_id + 1;
      table_ctr = table_id + 1;
    }
#if 0    /* 1 */
    temp_strm ≪ "'table_ctr'␣==␣" ≪ table_ctr ≪ endl ≪ "'table_id'␣==␣" ≪ table_id;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
  }    /* if (table_ctr ≡ 0) */
```

551.

Log

[LDF 2006.09.04.] !! BUG FIX: Added this section. The way it was before failed when there was more than one entry for “main_canonical_title” in a line, or more than one lines for Pica+ 021A.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else {
    table_id = table_ctr++;
  }
```

552. Write data to **Main_Titles** table. [LDF 2006.07.27.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  sql_strm ≪ "set␣identity_insert␣Main_Titles␣on␣" ≪ "insert␣Main_Titles␣" ≪ "(" ≪
      column_name ≪ column_strm.str( ) ≪ ")␣values␣(" ≪ table_id ≪ value_strm.str( ) ≪ ")␣" ≪
      "set␣identity_insert␣" ≪ main_table_name ≪ "␣off";
#if 0     /* 1 */
  temp_strm ≪ "sql_strm.str()␣==␣\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }     /* catch */
  sql_strm.str(""); }     /* if (Creating a new entry in Main_Titles) */
```

553. An identical entry already exists in **Main_Titles**. [LDF 2006.07.27.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  else {
    table_id = temp_ids.m_temp_id;
#if 0     /* 1 */
    temp_strm ≪ "An␣identical␣entry␣in␣‘Main_Titles’␣already␣exists." ≪ endl ≪
        "‘table_id’␣==␣" ≪ table_id;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
  }
```

554. Check whether there's already an entry for this record and main title in **Records_Main_Titles**. [LDF 2006.07.27.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  temp_ids.Close();
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣record_id␣" ≪ "from␣" ≪
      assoc_table_name ≪ "␣where␣record_id␣=␣" ≪ record_id ≪ "␣and␣" ≪ column_name ≪ "␣=␣" ≪
      table_id;
#if 0     /* 1 */
  temp_strm ≪ "SQL␣query:\n" ≪ sql_strm.str().c_str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣`cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }     /* catch */
  sql_strm.str("");
  temp_ids.Open();
  if (temp_ids.IsBOF() ∨ temp_ids.m_temp_id ≡ 0) {
#if 0     /* 1 */
    temp_strm ≪ "record_id␣==␣" ≪ record_id ≪ ",␣" ≪ column_name ≪ "␣=␣" ≪ table_id ≪ endl ≪
        "Creating␣an␣entry␣in␣`" ≪ assoc_table_name ≪ "'.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    sql_strm ≪ "insert␣" ≪ assoc_table_name ≪ "␣(record_id,␣" ≪ column_name ≪ ")␣values␣(" ≪
        record_id ≪ ",␣" ≪ table_id ≪ ")";
#if 0     /* 1 */
    temp_strm ≪ "SQL␣command:\n" ≪ sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    try {
      database→ExecuteSQL(sql_strm.str().c_str());
    }
    catch(CDBException *e)
    {
      temp_strm ≪ "Exception␣when␣calling␣`cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
          e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
          endl ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
      e→Delete();
    }    /* catch */
    sql_strm.str("");
  }    /* if (Create an entry in association table) */
  else    /* Entry already exists in association table */
  {
#if 0    /* 1 */
    temp_strm << "record_id␣==␣" << record_id << ",␣" << column_name << "␣=␣" << table_id << endl <<
        "An␣entry␣in␣‘" << assoc_table_name << "’␣already␣exists.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
  }    /* else (Entry already exists in association table) */
  temp_ids.Close();
  return table_ctr; }    /* End of Category_Container::sub_titles_category_func definition. */
```

555. publishers_database_providers_func. [LDF 2006.08.14.]

Log

[LDF 2006.08.14.] Added this function. It's called in **Category_Container**::`F_033A`.

```
⟨Declare Category_Container functions 426⟩ +≡
  static 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);
```

556.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::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)
      {
      stringstream temp_strm;
      stringstream sql_strm;
      Temp_IDs temp_ids;
#if 0     /* 1 */
      temp_strm << "Entering␣'Category_Container::publishers_database_providers_func'.";
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      string table;
      string assoc_table;
      string id_column;
      string name_column;
      if (table_switch) {
        table = "Publishers";
        assoc_table = "Records_Publishers";
        id_column = "publisher_id";
        name_column = "publisher_name";
      }
      else {
        table = "Database_Provider";
        assoc_table = "Records_Database_Provider";
        id_column = "database_provider_id";
        name_column = "database_provider_name";
      }
      sql_strm << "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣" << id_column << "␣from␣" << table <<
          "␣where␣" << name_column << "␣=␣'" << column_str << "'␣" << "and␣place␣=␣'" <<
          place_str << "'␣" << "and␣primary_info_source␣=␣";
      if (primary_switch) sql_strm << "1";
      else sql_strm << "0";
#if 0     /* 1 */
      temp_strm << "SQL␣Code:\n" << sql_strm.str() << endl;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      try {
        database->ExecuteSQL(sql_strm.str().c_str());
      }
      catch(CDBException *e)
      {
        temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
            e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
            endl ≪ sql_strm.str( );
        AfxMessageBox(temp_strm.str( ).c_str( ));
        temp_strm.str("");
        e→Delete( );
      }    /* catch */
      sql_strm.str("");
      temp_ids.Open( );
```

557. No corresponding entry in the **Publishers** or **Database_Providers** table. Create a new one. [LDF 2006.08.14.]

```
⟨ Define Category_Container functions 427 ⟩ +≡
  if (temp_ids.IsBOF( ) ∨ temp_ids.m_temp_id ≡ 0) {
#if 0    /* 1 */
  temp_strm ≪ "No␣corresponding␣entry␣in␣" ≪ table ≪ ".␣␣Creating␣a␣new␣one." ≪ endl;
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  sql_strm ≪ "insert␣" ≪ table ≪ "␣(" ≪ name_column ≪ ",␣place,␣primary_info_source)␣" ≪
      "values␣('" ≪ column_str ≪ "',␣'" ≪ place_str ≪ "',␣" ≪ static_cast⟨int⟩(primary_switch) ≪
      ")";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( ) ≪ endl;
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_strm.str("");
```

558. Get the *publisher_id* or *database_provider_id* value for the entry we've just created. [LDF 2006.08.14.]

Log

[LDF 2006.09.01.] !! BUG FIX: Now using *top* 1 in the *select* command contained in the SQL code.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣top␣1␣" ≪ id_column ≪ "␣from␣" ≪
      table ≪ "␣order␣by␣" ≪ id_column ≪ "␣desc";
#if 0    /* 1 */
  temp_strm ≪ "Getting␣‘" ≪ id_column ≪ "’␣value␣for␣new␣entry.\n" ≪ "SQL␣Code:\n" ≪
      sql_strm.str() ≪ endl;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  temp_ids.Close();
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }    /* catch */
  sql_strm.str("");
  temp_ids.Open();
#if 0    /* 1 */
  temp_strm ≪ "‘temp_ids.m_temp_id’␣==␣" ≪ temp_ids.m_temp_id ≪ endl;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  sql_strm ≪ "insert␣" ≪ assoc_table ≪ "␣(record_id,␣" ≪ id_column ≪ ")␣" ≪ "values␣(" ≪
      record_id ≪ ",␣" ≪ temp_ids.m_temp_id ≪ ")";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str() ≪ endl;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e->Delete();
  }   /* catch */
  sql_strm.str("");
  temp_ids.Close(); }   /* if */
```

559. A corresponding entry in the **Publishers** or **Database_Providers** table already exists. [LDF 2006.08.14.]

⟨ Define **Category_Container** functions 427 ⟩ +≡

```
  else   /* Entry in Publishers or Database_Providers already exists. */
  {
#if 0   /* 1 */
    temp_strm ≪ "A␣corresponding␣entry␣exists.␣␣‘temp_ids.m_temp_id’␣==␣" ≪
        temp_ids.m_temp_id ≪ endl;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    sql_strm ≪ "if␣not␣exists␣(select␣*␣from␣" ≪ assoc_table ≪ "␣where␣record_id␣=␣" ≪
        record_id ≪ "␣and␣" ≪ id_column ≪ "␣=␣" ≪ temp_ids.m_temp_id ≪ ")␣" ≪ "insert␣" ≪
        assoc_table ≪ "␣(record_id,␣" ≪ id_column ≪ ")␣" ≪ "values␣(" ≪ record_id ≪ ",␣" ≪
        temp_ids.m_temp_id ≪ ")";
#if 0   /* 1 */
    temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str() ≪ endl;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    try {
      database->ExecuteSQL(sql_strm.str().c_str());
    }
    catch(CDBException *e)
    {
      temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
          e->m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e->m_strError ≪ endl ≪ "SQL␣Code:" ≪
          endl ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
      e->Delete();
    }   /* catch */
    sql_strm.str("");
  }   /* else (Entry in Publishers or Database_Providers already exists.) */
  if (temp_ids.IsOpen()) temp_ids.Close();
  return 0; }   /* End of Category_Container::publishers_database_providers_func definition. */
```

560. Showing. [LDF Undated.]

⟨ Declare **Category_Container** functions 426 ⟩ +≡

```
  int show(stringstream &local_strm, string prefix = "");
```

561.

```
⟨ Define Category_Container functions 427 ⟩ +≡
  int Category_Container::show(stringstream &local_strm, string prefix)
  {
    local_strm ≪ prefix ≪ "‘pica_plus_category_id’␣==␣" ≪ pica_plus_category_id ≪ endl ≪
        "‘repeat_code’␣==␣" ≪ repeat_code ≪ endl ≪ "‘pica_3_category_id’␣==␣" ≪
        pica_3_category_id ≪ endl ≪ "‘content_description_german’␣==␣" ≪
        content_description_german ≪ endl ≪ "‘content_description_english’␣==␣" ≪
        content_description_english ≪ endl;
    size_t s = subcategory_vector.size();
    if (s ≤ 0) {
      local_strm ≪ "No␣subcategories␣on␣‘subcategory_vector’.";
    }
    else {
      local_strm ≪ static_cast⟨unsigned int⟩(s);
      if (s ≡ 1) local_strm ≪ "␣subcategory:\n";
      else local_strm ≪ "␣subcategories:\n";
      for (Subcategory_Vector_Type::iterator iter = subcategory_vector.begin();
            iter ≠ subcategory_vector.end(); ++iter) {
        iter→second→show(local_strm);
        local_strm ≪ endl;
      }    /* for */
    }    /* else */
    local_strm ≪ endl ≪ "**************************" ≪ endl ≪ endl;
    return 0;
  }    /* End of Category_Container::show definition. */
```

562. Putting Category_Container together. [LDF Undated.]

563. This is what gets written to `ctgcntnr.h`. [LDF Undated.]

```
⟨ ctgcntnr.h 563 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ “Using” declarations for namespaces 387 ⟩
  ⟨ Declare class Category_Container 423 ⟩
  ⟨ Type definitions 424 ⟩
```

564. This is what gets compiled. [LDF Undated.]

```
⟨ Include files 13 ⟩
⟨ ctgcntnr.web 418 ⟩
⟨ Declare static variables 422 ⟩
⟨ Define Category_Container functions 427 ⟩
```

565. Subcategory_Container (sbctgcnt.web). [LDF 2006.09.19.]

Log

[LDF 2006.07.20.] Created the file Subcategory_Container.h. Removed the declaration of **Subcategory_Container** from the file Category_Container.h and put it in the file Subcategory_Container.h.

[LDF 2006.09.19.] Created this file (sbctgcnt.web). It contains the code from Subcategory_Container.h and Subcategory_Container.cpp, and replaces these files.

```
⟨ sbctgcnt.web 565 ⟩ ≡
  static char id_string[] = "$Id:␣sbctgcnt.web,v␣1.9␣2006/11/27␣10:54:41␣Administrator␣Exp␣$";
```

This code is cited in sections 6 and 8.

This code is used in section 763.

566. Preprocessor macro calls. [LDF Undated.]

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

567. "Using" declarations for namespaces. [LDF 2006.09.19.]

```
⟨ "Using" declarations for namespaces 387 ⟩ +≡
  using namespace std;
```

568. Include files. [LDF Undated.]

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

569. Subcategory_Container class declaration. [LDF Undated.]

Log

[LDF 2006.07.31.] Made the data members **protected** rather than **public**. This made it necessary to add the **friend** declarations for the classes **Pica_Record** and **ZClient**.

[LDF 2006.09.06.] Added the **ofstream** &*log_strm* argument to all of the **static** functions that handle **Subcategory_Container** objects.

```
⟨ Declare class Subcategory_Container 569 ⟩ ≡
  class Subcategory_Container
  {
    friend class Pica_Record;
    friend class ZClient;
  protected: char pica_plus_field_id;
    string content_description_german;
    string content_description_english;
    string field_value;
    vector⟨Database_Command ∗⟩ database_commands;
  public: ⟨ Declare Subcategory_Container functions 572 ⟩
  };
```

This code is used in section 762.

570. Type definitions (typedefs). [LDF Undated.]

```
  format Subcategory_Map_Type Subcategory_Container
  format Subcategory_Vector_Type Subcategory_Map_Type
⟨ Type definitions 424 ⟩ +≡
  typedef map⟨char, Subcategory_Container ∗⟩ Subcategory_Map_Type;
  typedef vector⟨pair⟨char, Subcategory_Container ∗⟩⟩ Subcategory_Vector_Type;
```

571. Functions. [LDF 2006.09.19.]

572. Assignment operator. [LDF 2006.09.19.]

Log

LDF 2006.07.20. Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ ≡
  void operator=(const Subcategory_Container &);
```

See also sections 576, 581, 583, 586, 591, 612, 615, 626, 628, 630, 632, 634, 641, 644, 646, 651, 656, 658, 660, 663, 665, 667, 669, 672, 674, 676, 679, 681, 683, 685, 687, 690, 692, 695, 697, 700, 705, 708, 710, 712, 714, 717, 719, 723, 728, 733, 735, 737, 739, 742, 745, 747, 749, 751, 753, 755, 757, and 759.

This code is used in section 569.

573.

⟨ Define **Subcategory_Container** functions 573 ⟩ ≡

```
void Subcategory_Container::operator=(const Subcategory_Container &s)
{
  pica_plus_field_id = s.pica_plus_field_id;
  content_description_english = s.content_description_english;
  content_description_german = s.content_description_german;
}
```

See also sections 577, 578, 579, 582, 584, 587, 588, 589, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 613, 616, 617, 618, 619, 620, 621, 622, 623, 624, 627, 629, 631, 633, 635, 636, 637, 638, 639, 642, 645, 647, 648, 649, 650, 652, 653, 654, 655, 657, 659, 661, 664, 666, 668, 670, 673, 675, 677, 680, 682, 684, 686, 688, 691, 693, 696, 698, 701, 702, 703, 706, 709, 711, 713, 715, 718, 720, 724, 725, 726, 729, 730, 731, 734, 736, 738, 740, 743, 746, 748, 750, 752, 754, 756, 758, and 760.

This code is used in section 763.

574. Functions for Subcategories (PICA Fields). These functions either write information to PICA database, or store it in the **vector⟨Database_Command ∗⟩** *category→database_command_arguments* for processing by the **Category_Container** function for the current PICA category. Each of these functions has a **Category_Container** ∗*category* argument. [LDF 2006.07.25.] [LDF 2006.09.21.]

Log

[LDF 2006.07.25.] Added this section.

[LDF 2006.09.06.] Added the **ofstream** &*log_strm* argument to all of the **static** functions that handle **Subcategory_Container** objects.

[LDF 2006.09.06.] Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

575. Identifier and Date of the Original Catalogue Entry (Pica+ 001A / Pica3 0200). Kennung und Datum der Ersterfassung. [LDF Undated.]

576. Identifier and Date of the Original Catalogue Entry (‘0’). Kennung und Datum der Ersterfassung (‘0’). [LDF Undated.]

⟨ Declare **Subcategory_Container** functions 572 ⟩ +≡

```
static int f_001A_0(
CDatabase *database,
long record_id,
Category_Container *category,
Subcategory_Container *subcategory,
Output_Stream_Type &log_strm);
```

577.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_001A_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){
      stringstream temp_strm;
      stringstream sql_strm;
#if 0    /* 1 */
      temp_strm ≪ "Entering␣'Subcategory_Container::f_001A_0':" ≪ endl ≪
          "'record_id'␣==␣" ≪ record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪
          subcategory→field_value;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      sql_strm ≪ "update␣Records␣set␣date_original_entry␣=␣" ≪ "convert(datetime,␣'" ≪
          subcategory→field_value.substr(5) ≪ "',␣5),␣" ≪ "eln_original_entry␣=␣" ≪
          subcategory→field_value.substr(0, 4) ≪ "␣where␣record_id␣=␣" ≪ record_id;
#if 0    /* 1 */
      AfxMessageBox(sql_strm.str().c_str());
#endif
```

578.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }    /* catch */
  sql_strm.str("");
```

579.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
#if 0    /* 1 */
  temp_strm ≪ "Exiting␣'Subcategory_Container::f_001A_0'.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  return 0; }    /* End of Subcategory_Container::f_001A_0 definition. */
```

580. Identifier and Date of the Most Recent Change (Pica+ 001B / Pica3 0210). Kennung und Datum der letzten Änderung. [LDF 2006.08.03.]

581. Identifier and Date of the Change (‘0’).
Kennung und Datum der Änderung (‘0’). [LDF 2006.08.03.]

Log

[LDF 2006.08.03.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_001B_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

582.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_001B_0(CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_001B_0’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }     /* End of Subcategory_Container::f_001B_0 definition. */
```

583. Time (‘t’).
Uhrzeit (‘t’). [LDF 2006.08.03.]

Log

LDF 2006.08.03. Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_001B_t(CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

584.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_001B_t(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    temp_strm ≪ "In␣‘Subcategory_Container::f_001B_t’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
#if 0   /* 1 */
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }   /* End of Subcategory_Container::f_001B_t definition. */
```

585. Identifier and Date of the Change in Status (Pica+ 001D / Pica3 0230).
Kennung und Datum der Statusänderung

586. Identifier and Date of the Change (‘0’).
Kennung und Datum der Änderung.

Log

[LDF 2006.08.03.] Added this function.

[LDF 2006.08.29.] Added error-handling code for the case that the date string is “99-99-99”. Also calling **CDatabase**::*ExecuteSQL* in a **try** block. However, the code in the **catch** block doesn’t do anything yet.

To Do

Put real code in the **catch** block. Add similar error-handling code to the other functions that call the SQL *convert* function (?? or procedure, or whatever it is).

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_001D_0(CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

587.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container :: f_001D_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){
      stringstream temp_strm;
      stringstream sql_strm;
#if 0   /* 1 */
      temp_strm ≪ "Entering␣‘Subcategory_Container::f_001D_0’:␣" ≪ endl ≪
          "‘record_id’␣==␣" ≪ record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪
          subcategory→field_value;
      AfxMessageBox(temp_strm.str().c_str());
#endif
      temp_strm.str("");
      sql_strm ≪ "update␣Records␣set␣date_status_change␣=␣";
      if (subcategory→field_value.substr(5,8) ≡ "99-99-99") sql_strm ≪ "NULL,␣";
      else sql_strm ≪ "convert(datetime,␣'" ≪ subcategory→field_value.substr(5) ≪ "',␣5),␣";
      sql_strm ≪ "eln_status_change␣=␣" ≪ subcategory→field_value.substr(0,
          4) ≪ "␣where␣record_id␣=␣" ≪ record_id;
#if 0   /* 1 */
      temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

588.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }   /* catch */
  sql_strm.str("");
```

589.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
#if 0   /* 1 */
  temp_strm ≪ "Exiting␣‘Subcategory_Container::f_001D_0’.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  return 0; }    /* End of Subcategory_Container::f_001D_0 definition. */
```

590. Bibliographical Type and Status (Pica+ 002@ / Pica3 0500).
Bibliographische Gattung und Status. [LDF 2006.08.03.]

591. Bibliographical Type and Status (‘0’).
Bibliographische Gattung und Status (‘0’). [LDF 2006.08.03.]

Log

[LDF 2006.08.03.] Added this function.

[LDF 2006.09.01.] Made **stringstream** *sql_1_strm* local to this function. Formerly, it was **static** and local to this file.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_002_AT_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

592.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_002_AT_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){
      stringstream temp_strm;
      stringstream sql_strm;
#if 0   /* 1 */
      temp_strm ≪ "In␣‘Subcategory_Container::f_002_AT_0’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
          record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      stringstream sql_1_strm;
      Temp_IDs temp_ids;
```

593. Error handling: *subcategory→field_value.size*( ) < 3. [LDF 2006.08.03.]

To Do

[LDF 2006.08.03.] Add error log and write messages to it.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  if (subcategory→field_value.size( ) < 3) {
    temp_strm ≪ "ERROR!␣␣In␣‘Subcategory_Container::f_002_AT_0’:" ≪ endl ≪
        "There␣should␣be␣at␣least␣3␣characters␣in␣‘subcategory->field_value’,␣" ≪
        endl ≪ "but␣‘subcategory->field_value.size()’␣==␣" ≪ static_cast⟨unsigned
        int⟩(subcategory→field_value.size( )) ≪ "." ≪ endl ≪ "Not␣updating␣the␣‘B\
        ibliographic_Types’␣and␣‘Records_Bibliographic_Types’␣" ≪ "database␣tables." ≪
        endl ≪ "Exiting␣function␣unsuccessfully␣with␣return␣value␣1,␣and␣continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    return 1;
  }   /* if (subcategory→field_value.size( ) < 3) */
```

594.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣bibliographic_type_id␣" ≪
      "from␣Bibliographic_Types␣where␣" ≪ "physical_form␣=␣’" ≪ subcategory→field_value[0] ≪
      "’␣and␣bibliographic_representation␣=␣’" ≪ subcategory→field_value[1] ≪
      "’␣and␣description_status␣=␣’" ≪ subcategory→field_value[2] ≪ "’";
  sql_1_strm ≪ "insert␣Bibliographic_Types␣values␣(’" ≪ subcategory→field_value[0] ≪ "’,␣’" ≪
      subcategory→field_value[1] ≪ "’,␣’" ≪ subcategory→field_value[2] ≪ "’,␣";
```

595.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  if (subcategory→field_value.size( ) > 3) {
    sql_strm ≪ "␣and␣miscellaneous␣=␣’" ≪ subcategory→field_value[3] ≪ "’";
    sql_1_strm ≪ "’" ≪ subcategory→field_value[3] ≪ "’,␣";
    if (subcategory→field_value.size( ) > 4) {
      sql_strm ≪ "␣and␣bibliographic_representation_refinement␣=␣’" ≪
          subcategory→field_value[4] ≪ "’";
      sql_1_strm ≪ "’" ≪ subcategory→field_value[4] ≪ "’,␣";
      if (subcategory→field_value.size( ) > 5) {
        sql_strm ≪ "␣and␣transliteration_code␣=␣’" ≪ subcategory→field_value[5] ≪ "’";
        sql_1_strm ≪ "’" ≪ subcategory→field_value[5] ≪ "’";
      }
      else {
        sql_strm ≪ "␣and␣transliteration_code␣is␣NULL";
        sql_1_strm ≪ "NULL";
      }
    }   /* ¿ 4 */
    else {
      sql_strm ≪ "␣and␣bibliographic_representation_refinement␣is␣NULL";
      sql_1_strm ≪ "NULL,␣NULL";
    }
  }   /* ¿ 3 */
```

596.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  else {
    sql_strm ≪ "␣and␣miscellaneous␣is␣NULL";
    sql_1_strm ≪ "NULL,␣NULL,␣NULL";
  }
  sql_1_strm ≪ ")";
#if 0      /* 1 */
  temp_strm ≪ "SQL␣code:\n" ≪ sql_strm.str() ≪ endl ≪ "SQL␣1␣Code:\n" ≪ sql_1_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

597.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }      /* catch */
  sql_strm.str("");
```

598.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  temp_ids.Open();
```

599. No corresponding entry in the **Bibliographic_Types** table. We create one. [LDF 2006.08.03.]

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  if (temp_ids.IsBOF() ∨ temp_ids.m_temp_id ≡ 0) {
#if 0      /* 1 */
  temp_strm ≪ "No␣corresponding␣entry␣in␣'Bibliographic_Types'.␣␣Creating␣one.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

600.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_1_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_1_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_1_strm.str("");
```

601. Create entry in **Records_Bibliographic_Types**. [LDF Undated.]

Log

[LDF 2006.09.01.] !! BUG FIX: Now using *top*1 in the *select* command contained in the SQL code.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  temp_ids.Close( );
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣" ≪ "select␣top␣1␣bibliographic_type_id␣from\
      ␣␣Bibliographic_Types␣" ≪ "order␣by␣bibliographic_type_id␣desc";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
```

602.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_strm.str("");
```

603.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  temp_ids.Open( );
  if (temp_ids.IsBOF( ) ∨ temp_ids.m_temp_id ≡ 0) {
    temp_strm ≪ "ERROR!␣␣In␣‘Subcategory_Container::f_002_AT_0’:" ≪ endl ≪
        "Failed␣to␣retrieve␣the␣most␣recent␣‘bibliographic_type_id’␣from␣" ≪
        "the␣‘Bibliographic_Types’␣table." ≪ endl ≪
        "Exiting␣function␣unsuccesfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_ids.Close( );
    return 1;
  }    /* if */
#if 0    /* 1 */
  temp_strm ≪ "‘temp_ids.m_temp_id’␣==␣" ≪ temp_ids.m_temp_id;
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  sql_strm ≪ "insert␣Records_Bibliographic_Types␣(record_id,␣bibliographic_type_id)␣" ≪
      "values␣(" ≪ record_id ≪ ",␣" ≪ temp_ids.m_temp_id ≪ ")";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
```

604.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_strm.str("");
```

605.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  temp_ids.Close( ); }    /* if (No corresponding entry in the Bibliographic_Types table.) */
```

606.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  else {
#if 0    /* 1 */
  temp_strm ≪ "There␣is␣a␣corresponding␣entry␣in␣'Bibliographic_Types'." ≪ endl ≪
      "'temp_ids.m_temp_id'␣==␣" ≪ temp_ids.m_temp_id;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  sql_1_strm.str("");
  long bib_id = temp_ids.m_temp_id;
  temp_ids.Close();
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣record_id␣" ≪
      "from␣Records_Bibliographic_Types␣" ≪ "where␣record_id␣=␣" ≪ record_id ≪
      "␣and␣bibliographic_type_id␣=␣" ≪ bib_id;
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

607.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }    /* catch */
  sql_strm.str("");
```

608.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  temp_ids.Open(); if (temp_ids.IsBOF() ∨ temp_ids.m_temp_id ≡ 0) { temp_strm ≪
      "No␣corresponding␣entry␣in␣'Records_Bibliographic_Types'.␣␣Creating␣one.\n";
  sql_strm ≪ "insert␣Records_Bibliographic_Types␣(record_id,␣bibliographic_type_id)␣" ≪
      "values␣(" ≪ record_id ≪ ",␣" ≪ bib_id ≪ ")";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:␣␣" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

609.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_strm.str(""); }     /* if (Create new entry in Records_Bibliographic_Types table.) */
```

610. A corresponding entry already exists in the **Records_Bibliographic_Types** table. This normally shouldn’t occur. [LDF 2006.08.03.]

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  else {
#if 0    /* 1 */
    temp_strm ≪ "There␣is␣a␣corresponding␣entry␣in␣" ≪
        "‘Records_Bibliographic_Types’.␣␣Not␣creating␣one.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
  }
  temp_ids.Close( ); }     /* else */
  if (temp_ids.IsOpen( )) temp_ids.Close( );
  return 0; }     /* End of Subcategory_Container::f_002_AT_0 definition. */
```

611. Identification Number (PPN) (Pica+ 003@ / Pica3 0100).
Identifikationsnummer (PPN) (Pica+ 003@ / Pica3 0100). [LDF 2006.07.24.]
No information sheet for this category in the “Categorization Guidelines” (“Katagorisierungsrichtlinien”)! [LDF 2006.07.24.]
“PPN” = “PICA Production Number”. [LDF 2006.08.23.]

612. Identification Number (PPN) (‘0’).
Identifikationsnummer (PPN) (‘0’). [LDF 2006.07.24.]
No information sheet for this category in the “Categorization Guidelines” (“Katagorisierungsrichtlinien”)! [LDF 2006.07.24.]

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_003_AT_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

613.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_003_AT_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    stringstream sql_strm;
    temp_strm ≪ "In␣`Subcategory_Container::f_003_AT_0':␣" ≪ endl ≪ "`record_id'␣==␣" ≪
        record_id ≪ endl ≪ "`subcategory->field_value'␣==␣" ≪ subcategory→field_value;
#if 0   /* 1 */
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    sql_strm ≪ "update␣Records␣set␣identification_number␣=␣" ≪ "'" ≪ subcategory→field_value ≪
        "'␣" ≪ "where␣record_id␣=␣" ≪ record_id;
#if 0   /* 1 */
    AfxMessageBox(sql_strm.str().c_str());
#endif
    try {
      database→ExecuteSQL(sql_strm.str().c_str());
    }
    catch(CDBException *e)
    {
      temp_strm ≪ "Exception␣when␣calling␣`cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
          e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
          endl ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
      e→Delete();
    }   /* catch */
    sql_strm.str("");
    return 0;
  }   /* End of Subcategory_Container::f_003_AT_0 definition. */
```

614. Code(s) for Languages (Pica+ 010@ / Pica3 1500).
Code(s) für Sprachen (Pica+ 010@ / Pica3 1500). [LDF 2006.07.24.]

615. First Language Code for the Current Text (‘a’).
Erster Sprachencode für den vorliegenden Text (‘a’). [LDF 2006.08.03.]

Log

[LDF 2006.08.03.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_010_AT_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

616.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_010_AT_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){
      stringstream temp_strm;
      stringstream sql_strm;
#if 0    /* 1 */
      temp_strm ≪ "In␣‘Subcategory_Container::f_010_AT_a’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
          record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

617.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  Temp_IDs temp_ids;
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣language_id␣from␣Languages␣" ≪
      "where␣language_abbrev␣=␣’" ≪ subcategory→field_value ≪ "’";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code␣==␣" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

618.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }   /* catch */
  sql_strm.str("");
  temp_ids.Open( );
```

619. Error handling: No corresponding entry in the **Languages** table. All valid languages should have predefined entries. [LDF 2006.08.03.]

To Do

[LDF 2006.08.03.] Write error to error log. Perhaps make error log file a member of **ZClient**, and pass a **ZClient** * through the chain of functions.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  if (temp_ids.IsBOF( ) ∨ temp_ids.m_temp_id ≡ 0) {
    temp_strm ≪ "ERROR!␣␣In␣‘Subcategory_Container::f_010_AT_a’:" ≪ endl ≪
        "No␣‘Language’␣entry␣with␣‘language_abbrev’␣==␣" ≪ subcategory→field_value ≪ endl ≪
        "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    temp_ids.Close( );
    return 1;
  }   /* if */
```

620. A corresponding entry in the **Languages** table exists. [LDF 2006.08.03.]

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  else { long lang_id = temp_ids.m_temp_id;
#if 0    /* 1 */
  temp_strm ≪ "‘Language’␣entry␣" ≪ lang_id ≪ "␣has␣␣‘language_abbrev’␣==␣" ≪
      subcategory→field_value;
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  temp_ids.Close( );
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣record_id␣" ≪
      "from␣Records_Languages␣where␣record_id␣=␣" ≪ record_id ≪ "␣and␣language_id␣=␣" ≪
      lang_id ≪ "␣and␣association_type␣=␣’a’";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
```

621.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_strm.str("");
  temp_ids.Open( );
```

622.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  if (temp_ids.IsBOF() ∨ temp_ids.m_temp_id ≡ 0) {
#if 0     /* 1 */
  temp_strm ≪ "No␣corresponding␣entry␣in␣‘Records_Languages’.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  sql_strm ≪ "insert␣Records_Languages␣" ≪ "(record_id,␣language_id,␣association_ty\
      pe,␣association_type_name)␣" ≪ "values␣(" ≪ record_id ≪ ",␣" ≪ lang_id ≪
      ",␣'a',␣'Present␣Text')";
#if 0     /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

623.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }     /* catch */
  sql_strm.str(""); }     /* if */
```

624.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  else {
#if 0     /* 1 */
    temp_strm ≪ "There␣is␣a␣corresponding␣entry␣in␣‘Records_Languages’:\n" ≪
        "‘temp_ids.m_temp_id’␣==␣" ≪ temp_ids.m_temp_id;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
  }     /* else */
  temp_ids.Close(); }     /* else */
  if (temp_ids.IsOpen()) temp_ids.Close();
  return 0; }     /* End of Subcategory_Container::f_010_AT_a definition. */
```

625. Year of Appearance (Pica+ 011@ / Pica3 1100).
Erscheinungsjahr (Pica+ 011@ / Pica3 1100). [LDF 2006.08.04.]

626. Year of Appearance (Beginning), Form for Sorting (‘a’).
Erscheinungsjahr (Beginn), Sortierform (‘a’). [LDF 2006.08.04.]

This function removes non-digits from *subcategory→field_value* and checks that the remaining string contains exactly four characters. If it does, it writes it to the **year_appearance_begin** column for the current record in the **Records** table. If the string doesn't contain four characters, this function issues an error message and exits with return value 1. [LDF 2006.08.04.]

Log

[LDF 2006.08.04.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_011_AT_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

627.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_011_AT_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_011_AT_a’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    return year_appearance_func("year_appearance_begin", database, record_id, subcategory, log_strm);
  }    /* End of Subcategory_Container::f_011_AT_a definition. */
```

628. Year of Appearance (End), Form for Sorting (‘b’).
Erscheinungsjahr (Ende), Sortierform (‘b’). [LDF 2006.08.04.]

Log

[LDF 2006.08.04.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_011_AT_b(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

629.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_011_AT_b(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm << "In␣‘Subcategory_Container::f_011_AT_b’:␣" << endl << "‘record_id’␣==␣" <<
        record_id << endl << "‘subcategory->field_value’␣==␣" << subcategory->field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    return year_appearance_func("year_appearance_end", database, record_id, subcategory, log_strm);
  }    /* End of Subcategory_Container::f_011_AT_b definition. */
```

630. Original Year of Appearance (‘e’).
Ursprüngliches Erscheinungsjahr (‘e’). [LDF 2006.08.04.]

Log

[LDF 2006.08.04.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_011_AT_e(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

631.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_011_AT_e(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_011_AT_e’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    return year_appearance_func("year_appearance_original", database, record_id, subcategory, log_strm);
  }    /* End of Subcategory_Container::f_011_AT_e definition. */
```

632. Year of Appearance (according to RAK-WB) (‘n’).
Erscheinungsjahr (nach RAK-WB) (‘n’). [LDF 2006.08.04.]

Log

[LDF 2006.08.04.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_011_AT_n(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

633.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_011_AT_n(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_011_AT_n’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    return year_appearance_func("year_appearance_rak_wb", database, record_id, subcategory, log_strm);
  }    /* End of Subcategory_Container::f_011_AT_n definition. */
```

634. year_appearance_func. [LDF 2006.08.04.]

Log

[LDF 2006.08.04.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int year_appearance_func(
  string column,
  CDatabase *database,
  long record_id,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

635.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::year_appearance_func(string column, CDatabase *database, long
        record_id,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){
        stringstream temp_strm;
        stringstream sql_strm;
#if 0     /* 1 */
        temp_strm << "In␣‘Subcategory_Container::year_appearance_func’:␣" << endl <<
            "‘record_id’␣==␣" << record_id << endl << "‘subcategory->field_value’␣==␣" <<
            subcategory→field_value;
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
        string curr_str;
        for (unsigned int i = 0; i < subcategory→field_value.length(); ++i) {
          if (isdigit(subcategory→field_value[i])) curr_str += subcategory→field_value[i];
        }
#if 0     /* 1 */
        temp_strm << "‘curr_str’␣==␣" << curr_str;
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
```

636. Error handling: *curr_str.length*() ≠ 4. [LDF Undated.]

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  if (curr_str.length() ≠ 4) {
    temp_strm << "ERROR!␣␣In␣‘Subcategory_Container::year_appearance_func’:" <<
        endl << "‘record_id’␣==␣" << record_id << endl << "‘curr_str’␣==␣" <<
        curr_str << endl << "‘curr_str.length()’␣is␣not␣4." << endl <<
        "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    return 1;
  }    /* if (curr_str.length() ≠ 4) */
```

637.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  sql_strm << "update␣Records␣set␣" << column << "␣=␣" << curr_str << "␣where␣record_id␣=␣" <<
        record_id;
#if 0     /* 1 */
  temp_strm << "SQL␣Code:\n" << sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

638.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }   /* catch */
  sql_strm.str("");
```

639.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  return 0; }   /* End of Subcategory_Container :: year_appearance_func definition. */
```

640. General Material Name (Pica+ 016H / Pica3 1108).
Allgemeine Materialbenennung (Pica+ 016H / Pica3 1108). [LDF 2006.09.01.]

641. General Material Name (‘0’).
Allgemeine Materialbenennung (‘0’). [LDF 2006.09.01.]

Log

[LDF 2006.09.01.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_016H_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

642.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_016H_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
#if 0    /* 1 */
#define LDF_DEBUG
#else
#undef LDF_DEBUG
#endif
    stringstream temp_strm;
#ifdef LDF_DEBUG
    temp_strm ≪ "Entering␣‘Subcategory_Container::f_016H_0’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
#ifdef LDF_DEBUG
    temp_strm ≪ "Exiting␣‘Subcategory_Container::f_016H_0’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
#undef LDF_DEBUG
    return 0;
  }    /* End of Subcategory_Container::f_016H_0 definition. */
```

643. Main Canonical Title, Additional Information, Authorship (Pica+ 021A / Pica3 4000).

Hauptsachtitel, Zusätze, Verfasserangabe (Pica+ 021A / Pica3 4000). [LDF 2006.07.27.]

644. Standard Text (‘1’).
Standardtext (‘1’). [LDF 2006.07.27.]

Log

[LDF 2006.07.27.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_021A_1(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

645.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_021A_1(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm ≪ "Entering␣‘Subcategory_Container::f_021A_1’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    string curr_str;
    fix_string(subcategory→field_value, curr_str);
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, curr_str));
    return 0;
  }    /* End of Subcategory_Container::f_021A_1 definition. */
```

646. Main Canonical Title (‘a’).
Hauptsachtitel (‘a’). [LDF Undated.]

Log

[LDF Undated.] Added this function.

[LDF 2006.09.04.] !! BUG FIX: Added code for splitting long titles. These didn't cause errors when writing to the database, but they weren't entered completely. Errors then occurred when trying to read them from the database.

[LDF 2006.09.21.] !! BUG FIX: Actually, it turned out that the problem was the default limit for the length of the string stored in the **CStringA** data members of the classes derived from **CRecordset**. In this case, it was **class Main_Titles**. I've now fixed this for all of these classes.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_021A_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

647.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_021A_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){
      stringstream temp_strm;
#if 0     /* 1 */
      temp_strm ≪ "In␣‘Subcategory_Container::f_021A_a’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
          record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      string curr_str;
      fix_string(subcategory→field_value, curr_str);
#if 0     /* 1 */
      temp_strm ≪ "In␣‘Subcategory_Container::f_021A_a’:" ≪ endl ≪ "‘curr_str’␣==␣" ≪
          curr_str;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

648.

Log

[LDF 2006.09.11.] Increased the limit for the length of *curr_str* to 512.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  if (curr_str.length() ≤ 512)
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, curr_str));
  else { string temp_str;
  int char_ctr;
  int save_char_ctr;
  int continuation_ctr = 1;
  stringstream continuation_strm;
```

649.

Log

[LDF 2006.09.11.] Increased the limit for the length of *curr_str* to 512.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  while (curr_str.length() > 512) { save_char_ctr = char_ctr = static_cast⟨int⟩(min(curr_str.length() − 1,
       199));
#if 0     /* 1 */
  temp_strm ≪ "char_ctr␣==␣" ≪ char_ctr;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

650.

Log

[LDF 2006.09.05.] !! BUG FIX: No longer using *isspace*. Now testing for a space or tab character explicitly. *isspace* failed because of the numerical value of a character in a record. However, I wasn't able to determine the exact reason why this occurred. Also added the condition *char_ctr* > 0.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  while (¬(curr_str[char_ctr] ≡ '␣' ∨ curr_str[char_ctr] ≡ '\t') ∧ char_ctr > 0)  −−char_ctr;
  if (char_ctr ≡ 0)     /* If no spaces, just break at save_char_ctr. */
    char_ctr = save_char_ctr;
  temp_str = curr_str.substr(0, char_ctr);
#if 0     /* 1 */
  temp_strm ≪ "curr_str␣==␣'" ≪ curr_str ≪ "'" ≪ endl ≪ "temp_str␣==␣'" ≪ temp_str ≪ "'";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, temp_str));
  continuation_strm ≪ continuation_ctr++;
  category→database_command_arguments.push_back(make_pair('!', continuation_strm.str()));
  continuation_strm.str("");
  curr_str.erase(0, char_ctr);
#if 0     /* 1 */
  temp_strm ≪ "After␣erase:␣␣curr_str␣==␣'" ≪ curr_str ≪ "'" ≪ endl ≪
       "curr_str.length()␣==␣" ≪ curr_str.length();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  }     /* while */
  if (curr_str.length() > 0) {
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, curr_str));
    continuation_strm ≪ continuation_ctr++;
    category→database_command_arguments.push_back(make_pair('!', continuation_strm.str()));
  }
  }     /* else */
  return 0; }     /* End of Subcategory_Container::f_021A_a definition. */
```

651. Additions to the Main Canonical Title (‘d’).
Zusätze zum Hauptsachtitel. [LDF 2006.07.27.]

Log

[LDF 2006.07.27.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_021A_d(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

652.

Log

[LDF 2006.09.05.] !! BUG FIX: Added code for splitting long strings. These didn’t cause errors when writing to the database, but they weren’t entered completely. Errors then occurred when trying to read them from the database.

[LDF 2006.09.21.] !! BUG FIX: Actually, it turned out that the problem was the default limit for the length of the string stored in the **CStringA** data members of the classes derived from **CRecordset**. In this case, it was **class Main_Titles**. I’ve now fixed this for all of these classes.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_021A_d(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){
      stringstream temp_strm;
#if 0     /* 1 */
      temp_strm ≪ "In␣‘Subcategory_Container::f_021A_d’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
          record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      string curr_str;
      fix_string(subcategory→field_value, curr_str);
#if 0     /* 1 */
      temp_strm ≪ "In␣‘Subcategory_Container::f_021A_d’:" ≪ endl ≪ "‘curr_str’␣==␣" ≪
          curr_str;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

653.

Log

[LDF 2006.09.11.] Increased the limit for the length of *curr_str* to 512.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  if (curr_str.length() ≤ 512)
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, curr_str));
  else { string temp_str;
  int char_ctr;
  int save_char_ctr;
  int continuation_ctr = 1;
  stringstream continuation_strm;
```

654.

Log

[LDF 2006.09.11.] Increased the limit for the length of *curr_str* to 512.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  while (curr_str.length() > 512) { save_char_ctr = char_ctr = static_cast⟨int⟩(min(curr_str.length() − 1,
        199));
#if 0    /* 1 */
  temp_strm ≪ "char_ctr␣==␣" ≪ char_ctr ≪ "‘curr_str[" ≪ char_ctr ≪ "]’␣==␣" ≪
        curr_str[char_ctr];
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
```

655.

Log

[LDF 2006.09.05.] !! BUG FIX: No longer using *isspace*. Now testing for a space or tab character explicitly. *isspace* failed because of the numerical value of a character in a record. However, I wasn't able to determine the exact reason why this occurred. Also added the condition *char_ctr* > 0.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  while (¬(curr_str[char_ctr] ≡ '␣' ∨ curr_str[char_ctr] ≡ '\t')) −−char_ctr;
  if (char_ctr ≡ 0)    /* If no spaces, just break at save_char_ctr. */
    char_ctr = save_char_ctr;
  temp_str = curr_str.substr(0, char_ctr);
#if 0    /* 1 */
  temp_strm ≪ "curr_str␣==␣'" ≪ curr_str ≪ "'" ≪ endl ≪ "temp_str␣==␣'" ≪ temp_str ≪ "'";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, temp_str));
  continuation_strm ≪ continuation_ctr++;
  category→database_command_arguments.push_back(make_pair('?', continuation_strm.str()));
  continuation_strm.str("");
  curr_str.erase(0, char_ctr);
#if 0    /* 1 */
  temp_strm ≪ "After␣erase:␣␣curr_str␣==␣'" ≪ curr_str ≪ "'" ≪ endl ≪
      "curr_str.length()␣==␣" ≪ curr_str.length();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  }    /* while */
  if (curr_str.length() > 0) {
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, curr_str));
    continuation_strm ≪ continuation_ctr++;
    category→database_command_arguments.push_back(make_pair('?', continuation_strm.str()));
  }
  }    /* else */
  return 0; }    /* End of Subcategory_Container::f_021A_d definition. */
```

656. Additional Creator (‘e’).
Zu ergänzender Urheber (‘e’). [LDF 2006.07.27.]

Log

[LDF 2006.07.27.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_021A_e(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

657.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container :: f_021A_e(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    temp_strm ≪ "In␣‘Subcategory_Container::f_021A_e’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
#if 0    /* 1 */
    AfxMessageBox(temp_strm.str( ).c_str( ));
#endif
    temp_strm.str("");
    string curr_str;
    fix_string(subcategory→field_value, curr_str);
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, curr_str));
    return 0;
  }    /* End of Subcategory_Container :: f_021A_e definition. */
```

658. Parallel Canonical Title (‘f’).
Parallelsachtitel (‘f’). [LDF 2006.07.27.]

Log

[LDF 2006.07.27.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_021A_f(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

659.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_021A_f(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    temp_strm ≪ "In␣‘Subcategory_Container::f_021A_f’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
#if 0     /* 1 */
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    string curr_str;
    fix_string(subcategory→field_value, curr_str);
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, curr_str));
    return 0;
  }     /* End of Subcategory_Container::f_021A_f definition. */
```

660. Authorship (‘h’).
Verfasserangabe (‘h’). [LDF 2006.07.27.]

Log

[LDF 2006.07.27.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_021A_h(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

661.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_021A_h(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    temp_strm ≪ "In␣'Subcategory_Container::f_021A_h':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
#if 0     /* 1 */
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    string curr_str;
    fix_string(subcategory→field_value, curr_str);
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, curr_str));
    return 0;
  }    /* End of Subcategory_Container::f_021A_h definition. */
```

662. 028A.

663. ('9'). [LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028A_9(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

664.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_028A_9(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    temp_strm ≪ "In␣'Subcategory_Container::f_028A_a':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
#if 0    /* 1 */
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }    /* End of Subcategory_Container::f_028A_9 definition. */
```

665. ('a'). [LDF Undated.]

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028A_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

666.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_028A_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    temp_strm ≪ "In␣'Subcategory_Container::f_028A_a':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
#if 0    /* 1 */
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    string curr_surname;
    fix_string(subcategory→field_value, curr_surname);
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        curr_surname));
    return 0;
  }    /* Subcategory_Container::f_028A_a */
```

667. ('c'). [LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028A_c(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

668.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_028A_c(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    temp_strm ≪ "In␣‘Subcategory_Container::f_028A_c’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
#if 0    /* 1 */
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    return 0;
  }    /* End of Subcategory_Container::f_028A_c definition. */
```

669. (‘d’). [LDF Undated.]

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028A_d(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

670.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_028A_d(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    temp_strm ≪ "In␣'Subcategory_Container::f_028A_d':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value ≪ endl;
    string curr_given_name;
    fix_string(subcategory→field_value, curr_given_name);
    temp_strm ≪ "'curr_given_name'␣=␣" ≪ curr_given_name;
#if 0    /* 1 */
    AfxMessageBox(temp_strm.str().c_str());
#endif
    temp_strm.str("");
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        curr_given_name));
    return 0;
  }    /* End of Subcategory_Container::f_028A_d definition. */
```

671. Personal Name; Author (Pica+ 028B / Pica3 120, 3000–3009).
Personenname; Verfasser (Pica+ 028B / Pica3 120, 3000–3009). [LDF 2006.09.07.]

Pica+ 028B, Pica3 120, 3000–3009

Pica3 120:	Personal Name (Cataloguing form according to RSWK) (RSWK = Regeln für den Schlagwortkatalog = Rules for the Subject Catalogue)
Pica3 3000–3009:	Author
Pica3 120:	Personenname (Ansetzungsform nach RSWK) (RSWK = Regeln für den Schlagwortkatalog)
Pica3 3000–3009:	Verfasser

Log

[LDF 2006.09.07.] Added this section.

672. Identification Number (PPN) (‘9’).
Identifikationsnummer (PPN) (‘9’). [LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028B_9(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

673.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_028B_9(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm << "In␣‘Subcategory_Container::f_028B_9’:␣" << endl << "‘record_id’␣==␣" <<
        record_id << endl << "‘subcategory->field_value’␣==␣" << subcategory->field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category->database_command_arguments.push_back(make_pair(subcategory->pica_plus_field_id,
        subcategory->field_value));
    return 0;
  }    /* End of Subcategory_Container::f_028B_9 definition. */
```

674. Surname (‘a’).
Familienname (‘a’). [LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028B_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

675.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_028B_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_028B_a':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }     /* End of Subcategory_Container::f_028B_a definition. */
```

676. Given Name ('d').
Vorname ('d'). [LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028B_d(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

677.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_028B_d(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_028B_d’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }     /* End of Subcategory_Container::f_028B_d definition. */
```

678. 028C. [LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this section.

---

679. Expansion of the Cataloguing Form.
Expansion der Ansetzungsform. [LDF 2006.11.27.]
Database table: Contributors:expansion. [LDF 2006.11.27.]

Log

[LDF 2006.11.27.] Added this function. It currently has a dummy definition.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028C_8(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

680.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container :: f_028C_8 (
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_028C_8':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
#if 0    /* 1 */
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
#endif
    return 0;
  }    /* End of Subcategory_Container :: f_028C_8 definition. */
```

681. ('9'). [LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028C_9 (
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

682.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_028C_9(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm << "In␣‘Subcategory_Container::f_028C_9’:␣" << endl << "‘record_id’␣==␣" <<
        record_id << endl << "‘subcategory->field_value’␣==␣" << subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }    /* End of Subcategory_Container::f_028C_9 definition. */
```

683. (‘a’). [LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028C_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

684.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_028C_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_028C_a':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
         record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    string curr_surname;
    fix_string(subcategory→field_value, curr_surname);
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
         curr_surname));
    return 0;
  }    /* End of Subcategory_Container::f_028C_a definition. */
```

685. ('c'). [LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028C_c(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

686.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_028C_c(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_028C_c’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    return 0;
  }    /* End of Subcategory_Container::f_028C_c definition. */
```

687. (‘d’). [LDF 2006.07.26.]

Log

[LDF 2006.07.26.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_028C_d(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

688.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_028C_d(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_028C_d':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    string curr_given_name;
    fix_string(subcategory→field_value, curr_given_name);
#if 0     /* 1 */
    temp_strm ≪ "'curr_given_name'␣==␣" ≪ curr_given_name;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        curr_given_name));
    return 0;
  }     /* End of Subcategory_Container::f_028C_d definition. */
```

689. 029A. Primary corporate entity. Pica+ 029A / Pica3 3100; Primrkrperschaft. [LDF 2006.11.27.] ▌
The functions in this section are intended to write to the *primary_corporate_entity* and *primary_corporate_entity_expansio*
columns of the **Publishers** database table, which otherwise contains data from Pica+ 033A / Pica3 4030: "Ort, Verlag".

Log

[2006.11.27.] Added this section.

690. '8' Expansion of the cataloguing form. Expansion der Ansetzungsform. [LDF 2006.11.27.]

Log

[2006.11.27.] Added this function. It currently has a dummy definition.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_029A_8(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

691.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_029A_8(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_029A_8':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
#if 0     /* 1 */
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
#endif
    return 0;
  }     /* End of Subcategory_Container::f_029A_8 definition. */
```

692. 'a' Corporate entity. [LDF 2006.11.27.]

Log

[2006.11.27.] Added this function. It currently has a dummy definition.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_029A_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

693.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_029A_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm << "In␣‘Subcategory_Container::f_029A_a’:␣" << endl << "‘record_id’␣==␣" <<
        record_id << endl << "‘subcategory->field_value’␣==␣" << subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
#if 0    /* 1 */
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
#endif
    return 0;
  }    /* End of Subcategory_Container::f_029A_a definition. */
```

694. 033A. [LDF Undated.]

695. (‘n’). [LDF 2006.08.14.]

Log

[LDF 2006.08.14.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_033A_n(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

696.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_033A_n(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_033A_n':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→pica_plus_field_id ≪
        endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    string curr_str;
    fix_string(subcategory→field_value, curr_str);
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, curr_str));
    return 0;
  }     /* End of Subcategory_Container::f_033A_n definition. */
```

697. ('p'). [LDF 2006.08.14.]

Log

[LDF 2006.08.14.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_033A_p(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

698.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_033A_p(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_033A_p':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→pica_plus_field_id ≪
        endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    string curr_str;
    fix_string(subcategory→field_value, curr_str);
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id, curr_str));
    return 0;
  }     /* End of Subcategory_Container::f_033A_p definition. */
```

699. Size or Range, Specification of Material, Technical System (Pica+ 034D / Pica3 4060). Umfangsangabe, spezifische Materialbenennung, technisches System (Pica+ 034D / Pica3 4060) [LDF 2006.08.14.] ▮

700. Text ('a'). [LDF 2006.08.14.]

Log

[LDF 2006.08.14.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_034D_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

701.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_034D_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){
       stringstream temp_strm;
       stringstream sql_strm;
#if 0     /* 1 */
       temp_strm ≪ "In␣'Subcategory_Container::f_034D_a':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
           record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
       AfxMessageBox(temp_strm.str().c_str());
       temp_strm.str("");
#endif
       Temp_IDs temp_ids;
       sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣pica_category_id␣from␣" ≪
           "PICA_Categories␣where␣pica_3_category_code␣=␣4060";
#if 0     /* 1 */
       temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
       AfxMessageBox(temp_strm.str().c_str());
       temp_strm.str("");
#endif
       try {
         database→ExecuteSQL(sql_strm.str().c_str());
       }
       catch(CDBException *e)
       {
         temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
             e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
             endl ≪ sql_strm.str();
         AfxMessageBox(temp_strm.str().c_str());
         temp_strm.str("");
         e→Delete();
       }    /* catch */
       sql_strm.str("");
       temp_ids.Open();     /* ***** (5) Error handling: No entry in the PICA_Categories table for
           PICA3 4060, PICA+ 034D. LDF 2006.08.15. */
       if (temp_ids.IsBOF() ∨ temp_ids.m_temp_id ≡ 0) {
         temp_strm ≪ "ERROR!␣␣In␣'Subcategory_Container::f_034D_a':" ≪ endl ≪
             "No␣entry␣in␣the␣'PICA_Categories'␣table␣for␣" ≪ "PICA+␣034D." ≪ endl ≪
             "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
         AfxMessageBox(temp_strm.str().c_str());
         temp_strm.str("");
         temp_ids.Close();
         return 1;
       }    /* if */
       unsigned int pica_category_id = temp_ids.m_temp_id;
#if 0     /* 1 */
```

```
        temp_strm ≪ "'pica_category_id'␣==␣" ≪ pica_category_id;
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
        temp_ids.Close();      /* ***** (5) Get pica_field_id. [LDF 2006.08.15.]  */
        sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣pica_field_id␣from␣" ≪
            "PICA_Fields␣where␣pica_plus_field_code␣=␣'a'␣" ≪
            "and␣description_english␣=␣'Text'";
#if 0    /* 1 */
        temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
        try {
          database→ExecuteSQL(sql_strm.str().c_str());
        }
        catch(CDBException *e)
        {
          temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
              e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
              endl ≪ sql_strm.str();
          AfxMessageBox(temp_strm.str().c_str());
          temp_strm.str("");
          e→Delete();
        }    /* catch */
        sql_strm.str("");
        temp_ids.Open();      /* ***** (5) Error handling: No entry in the PICA_Fields table for 'a'
            (Text). LDF 2006.08.15. */
        if (temp_ids.IsBOF() ∨ temp_ids.m_temp_id ≡ 0) {
          temp_strm ≪ "ERROR!␣␣In␣'Subcategory_Container::f_034D_a':" ≪ endl ≪
              "No␣entry␣in␣the␣'PICA_Fields'␣table␣for␣" ≪ "'a'␣(Text)." ≪ endl ≪
              "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
          AfxMessageBox(temp_strm.str().c_str());
          temp_strm.str("");
          temp_ids.Close();
          return 1;
        }    /* if */
        int pica_field_id = temp_ids.m_temp_id;
#if 0    /* 1 */
        temp_strm ≪ "'pica_field_id'␣==␣" ≪ pica_field_id;
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
        temp_ids.Close();
        string curr_text;
        fix_string(subcategory→field_value, curr_text);
        sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣physical_description_id␣" ≪
            "from␣Physical_Descriptions␣where␣text␣=␣'" ≪ curr_text ≪ "'␣" ≪
            "and␣pica_category_id␣=␣" ≪ pica_category_id ≪ "␣and␣pica_field_id␣=␣" ≪ pica_field_id;
#if 0    /* 1 */
        temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
#endif
        try {
          database→ExecuteSQL(sql_strm.str().c_str());
        }
        catch(CDBException *e)
        {
          temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
              e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
              endl ≪ sql_strm.str();
          AfxMessageBox(temp_strm.str().c_str());
          temp_strm.str("");
          e→Delete();
        }    /* catch */
        sql_strm.str("");
        temp_ids.Open();
        unsigned int physical_description_id;
```

702. There’s no corresponding entry in **Physical_Descriptions** yet. [LDF 2006.08.15.]

Log

[LDF 2006.09.01.] !! BUG FIX: Now using *top* 1 in the *select* command contained in the SQL code.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  if (temp_ids.IsBOF() ∨ temp_ids.m_temp_id ≡ 0) {
#if 0    /* 1 */
  temp_strm ≪ "An␣entry␣in␣‘Physical_Descriptions’␣doesn’t␣exist.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  temp_ids.Close();
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣top␣1␣physical_description_id␣" ≪
      "from␣Physical_Descriptions␣order␣by␣physical_description_id␣" ≪ "desc";
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }   /* catch */
  sql_strm.str("");
  temp_ids.Open();
```

703. Error handling: Failed to get highest *physical_description_id* from **Physical_Descriptions** table. [LDF 2006.08.15.]

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  if (temp_ids.IsBOF()) {
    temp_strm ≪ "ERROR!␣␣In␣‘Subcategory_Container::f_034D_a’:" ≪
        endl ≪ "Failed␣to␣get␣the␣highest␣‘physical_description_id’␣" ≪
        "from␣the␣‘Physical_Descriptions’␣table." ≪ endl ≪
        "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    temp_ids.Close();
    return 1;
  }   /* if (temp_ids.IsBOF()) */
  physical_description_id = temp_ids.m_temp_id + 1;
  temp_ids.Close();
#if 0   /* 1 */
  temp_strm ≪ "The␣new␣‘physical_description_id’␣==␣" ≪ physical_description_id;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif   /* ****** (6) Insert a new line into Physical_Descriptions. LDF 2006.08.15. */
  sql_strm ≪ "set␣identity_insert␣Physical_Descriptions␣on␣" ≪
      "insert␣Physical_Descriptions␣(physical_description_id,␣" ≪
      "text,␣pica_category_id,␣pica_field_id)␣" ≪ "values␣(" ≪ physical_description_id ≪
      ",␣’" ≪ curr_text ≪ "’,␣" ≪ pica_category_id ≪ ",␣" ≪ pica_field_id ≪
      ")␣set␣identity_insert␣Physical_Descriptions␣off";
#if 0   /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }   /* catch */
  sql_strm.str("");
    /* ****** (6) Insert a new line into Records_Physical_Descriptions. LDF 2006.08.15. */
  sql_strm ≪ "insert␣Records_Physical_Descriptions␣" ≪
      "(record_id,␣physical_description_id)␣" ≪ "values␣(" ≪ record_id ≪ ",␣" ≪
      physical_description_id ≪ ")";
#if 0   /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
```

```
#endif
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }   /* catch */
  sql_strm.str(""); }   /* if (No corresponding entry in Physical_Descriptions yet.) */
    /* ***** (5) An entry in Physical_Descriptions already exists. LDF 2006.08.15. */
  else   /* A corresponding entry in Physical_Descriptions already exists. */
  {
    physical_description_id = temp_ids.m_temp_id;
#if 0   /* 1 */   /* 1 */
    temp_strm ≪ "An␣entry␣in␣'Physical_Descriptions'␣already␣exists." ≪ endl ≪
        "'physical_description_id'␣==␣" ≪ physical_description_id;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
  }   /* else (A corresponding entry in Physical_Descriptions already exists.) */
      /* ***** (5) Insert an entry into Records_Physical_Descriptions, unless one already exists.
        LDF 2006.08.15. */
  sql_strm ≪ "if␣not␣exists␣(select␣*␣from␣Records_Physical_Descriptions␣" ≪
      "where␣record_id␣=␣" ≪ record_id ≪ "␣and␣physical_description_id␣" ≪ "=␣" ≪
      physical_description_id ≪ ")␣" ≪ "insert␣Records_Physical_Descriptions␣" ≪
      "(record_id,␣physical_description_id)␣values␣" ≪ "(" ≪ record_id ≪ ",␣" ≪
      physical_description_id ≪ ")";
#if 0   /* 1 */   /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str().c_str());
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    e→Delete();
  }   /* catch */
  sql_strm.str("");
  if (temp_ids.IsOpen()) temp_ids.Close();
  return 0; }    /* End of Subcategory_Container::f_034D_a definition. */
```

704. Link To Superordinate Entity. Pica+ 039B / Pica3 4241. Verknüpfung zur größeren Einheit. [LDF 2006.11.23.]

Log

[2006.11.23.] Added this section.

---

705. Title of the Superordinate Entity ('a'). Pica+ 039B / Pica3 4241. Titel der größeren Einheit. [LDF 2006.11.23.]

Log

[2006.11.23.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_039B_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

706.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_039B_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_039B_a':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    return 0;
  }    /* End of Subcategory_Container::f_039B_a definition. */
```

707. Main Subject and Subsidiary Subjects (Subject Assignment). [LDF 2006.08.21.]

Pica+ 041A, Pica3 800, 5100–5199

Pica3 800:	Main Subject and Subsidiary Subjects (Subject Assignment)
Pica3 5100–5199	RSWK Chains (RSWK: Regeln für den Schlagwortkatalog = Rules for the Subject Catalogue)
Pica3 800:	Hauptschlagwort und Unterschlagwörter (Schlagwortansetzung)
Pica3 5100–5199	RSWK-Ketten (RSWK: Regeln für den Schlagwortkatalog)

Log

[LDF 2006.08.21.] Added this section.

---

708. Identification Number (PPN) ('9').
Identifikationsnummer (PPN) ('9'). [LDF 2006.08.21.]
This field is only present in Pica3 5100–5199, not Pica3 800. [LDF 2006.08.21.]

Log

[LDF 2006.08.21.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_041A_9(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

709.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_041A_9(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */     /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_041A_9’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }     /* End of Subcategory_Container::f_041A_9 definition. */
```

710. Main Subject/Subject (‘a’).
Second and Additional Permutation Pattern
Subject Chain Information

Hauptschlagwort/Schlagwort (‘a’)
Zweites und weiteres Permutationsmuster
Angaben zur Schlagwortkette [LDF 2006.08.21.]

This field is called “Hauptschlagwort” (“Main Subject”) in Pica3 800 and “Schlagwort” (“Subject”) in Pica3 5100–5199. [LDF 2006.08.21.]

In Pica3 5100–5199 it is ambiguous. It can have the additional meanings “Zweites und weiteres Permutationsmuster” (“Second and Additional Permutation Pattern”) and “Angaben zur Schlagwortkette” (“Subject Chain Information”). [LDF 2006.08.21.]

Log

[LDF 2006.08.21.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_041A_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

711.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_041A_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */     /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_041A_a’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }    /* End of Subcategory_Container::f_041A_a definition. */
```

712. Permutation Pattern (‘f’).
Permutationsmuster (‘f’). [LDF 2006.08.21.]
This field is only present in Pica3 5100–5199, not Pica3 800. [LDF 2006.08.21.]

Log

[LDF 2006.08.21.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_041A_f(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

713.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_041A_f(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */    /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_041A_f’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }    /* End of Subcategory_Container::f_041A_f definition. */
```

714. Indicator/Subject Indicator (‘S’).
Indikator/Schlagwortindikator (‘S’). [LDF 2006.08.21.]

This field is called “Indikator” (“Indicator”) in Pica3 800 and “Schlagwortindikator” (“Subject Indicator”) in Pica3 5100–5199. [LDF 2006.08.21.]

Log

[LDF 2006.08.21.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_041A_S(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

715.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_041A_S(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm << "In␣'Subcategory_Container::f_041A_S':␣" << endl << "'record_id'␣==␣" <<
        record_id << endl << "'subcategory->field_value'␣==␣" << subcategory->field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category->database_command_arguments.push_back(make_pair(subcategory->pica_plus_field_id,
        subcategory->field_value));
    return 0;
  }    /* End of Subcategory_Container::f_041A_S definition. */
```

716. 045F. DDC. Dewey Decimal Classification. Pica+ 045F / Pica3 5010. [LDF 2006.11.27.]

Log

[2006.11.27.] Added this section.

---

717. 'a' Notation. [LDF 2006.11.27.]

Log

[2006.11.27.] Added this function. It currently has a dummy definition.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_045F_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

718.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_045F_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0   /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_045F_a':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
#if 0   /* 1 */
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
#endif
    return 0;
  }   /* End of Subcategory_Container::f_045F_a definition. */
```

719. 'b' Additional identifier. Zusatzkennung. [LDF 2006.11.27.]

Log

[2006.11.27.] Added this function. It currently has a dummy definition.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_045F_b(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

720.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_045F_b(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0      /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_045F_b':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
#if 0      /* 1 */
    category→database_command_brguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
#endif
    return 0;
  }      /* End of Subcategory_Container::f_045F_b definition. */
```

721. 045H. Pica+ 045H / Pica3 5400. Synthetic DDC Notation. Synthetische DDC-Notation. (Field or fields unknown.)

To Do

[LDF 2006.11.27.] Find information about this category. There's no PDF file for it in the GBV's loose-leaf collection.

---

722. Content Summary (short) (Pica+ 047I / Pica3 4207).
Inhaltliche Zusammenfassung (kurz) (Pica+ 047I / Pica3 4207). [LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this section.

---

723. Text ('a'). [LDF 2006.09.07.]

Log

[LDF 2006.09.07.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_047I_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

724.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_047I_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){
      stringstream temp_strm;
      stringstream sql_strm;
#if 0    /* 1 */
      temp_strm ≪ "In␣‘Subcategory_Container::f_047I_a’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
          record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

725.

Log

[LDF 2006.09.11.] Now setting *string_length_limit* to 1024 rather than 256.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  unsigned short string_length_limit = 1024;
```

726. Find the value to use for **Content_Summaries** :: *content_summary_id*. [LDF 2006.09.07.]

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  Temp_IDs temp_ids;
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣top␣1␣content_summary_id␣" ≪
      "from␣Content_Summaries␣order␣by␣content_summary_id␣desc";
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_strm.str("");
  long content_summary_id;
  temp_ids.Open( );
  if (temp_ids.IsBOF( ) ∨ temp_ids.m_temp_id ≡ 0) content_summary_id = 1;
  else content_summary_id = temp_ids.m_temp_id + 1;
  temp_ids.Close( );
  string curr_str;
  fix_string(subcategory→field_value, curr_str);
  if (curr_str.length( ) ≤ string_length_limit) {
    sql_strm ≪ "set␣identity_insert␣Content_Summaries␣on␣" ≪
        "insert␣Content_Summaries␣(content_summary_id,␣" ≪
        "record_id,␣continuation,␣content_summary)␣" ≪ "values␣(" ≪
        content_summary_id ≪ ",␣" ≪ record_id ≪ ",␣" ≪ 0 ≪ ",␣'" ≪ curr_str ≪
        "')␣" ≪ "set␣identity_insert␣Content_Summaries␣off";
#if 0    /* 1 */
    temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    try {
      database→ExecuteSQL(sql_strm.str( ).c_str( ));
    }
    catch(CDBException *e)
    {
      temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
          e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
          endl ≪ sql_strm.str( );
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
      e→Delete( );
    }    /* catch */
    sql_strm.str("");
  }    /* if (curr_str.length( ) ≤ string_length_limit) */
  else    /* curr_str.length() > string_length_limit */
  {
#if 0    /* 1 */
    temp_strm ≪ "'curr_str.length()'␣==␣" ≪ static_cast⟨unsigned
        int⟩(curr_str.length()) ≪ "␣(␣>␣string_length_limit)";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    string temp_str;
    int char_ctr;
    int save_char_ctr;
    int continuation_ctr = 1;
    stringstream continuation_strm;
    while (curr_str.length() > string_length_limit) {
      save_char_ctr = char_ctr = static_cast⟨int⟩(min(static_cast⟨int⟩(curr_str.length()) − 1,
          (static_cast⟨int⟩(string_length_limit) − 1)));
      while (¬(curr_str[char_ctr] ≡ '␣' ∨ curr_str[char_ctr] ≡ '\t') ∧ char_ctr > 0)  −−char_ctr;
#if 0    /* 1 */
      temp_strm ≪ "char_ctr␣==␣" ≪ char_ctr;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      if (char_ctr ≡ 0)    /* If no spaces, just break at save_char_ctr. */
        char_ctr = save_char_ctr;
      temp_str = curr_str.substr(0, char_ctr);
      continuation_strm ≪ continuation_ctr++;
      sql_strm ≪ "set␣identity_insert␣Content_Summaries␣on␣" ≪
          "insert␣Content_Summaries␣(content_summary_id,␣" ≪
          "record_id,␣continuation,␣content_summary)␣" ≪ "values␣(" ≪
          content_summary_id++ ≪ ",␣" ≪ record_id ≪ ",␣" ≪ continuation_strm.str() ≪ ",␣'" ≪
          temp_str ≪ "')␣" ≪ "set␣identity_insert␣Content_Summaries␣off";
#if 0    /* 1 */
      temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      try {
        database→ExecuteSQL(sql_strm.str().c_str());
      }
      catch(CDBException *e)
      {
        temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
            e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
            endl ≪ sql_strm.str();
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
        e→Delete();
      }    /* catch */
      sql_strm.str("");
      continuation_strm.str("");
      curr_str.erase(0, char_ctr);
#if 0     /* 1 */
      temp_strm ≪ "After␣erase:␣␣curr_str␣==␣'" ≪ curr_str ≪ "'" ≪ endl ≪
          "curr_str.length()␣=␣" ≪ static_cast⟨unsigned int⟩(curr_str.length());
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
    }     /* while */
    if (curr_str.length() > 0) {
      continuation_strm ≪ continuation_ctr++;
      sql_strm ≪ "set␣identity_insert␣Content_Summaries␣on␣" ≪
          "insert␣Content_Summaries␣(content_summary_id,␣" ≪
          "record_id,␣continuation,␣content_summary)␣" ≪ "values␣(" ≪ content_summary_id ≪
          ",␣" ≪ record_id ≪ ",␣" ≪ continuation_strm.str() ≪ ",␣'" ≪ curr_str ≪ "')␣" ≪
          "set␣identity_insert␣Content_Summaries␣off";
#if 0     /* 1 */
      temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      try {
        database→ExecuteSQL(sql_strm.str().c_str());
      }
      catch(CDBException *e)
      {
        temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
            e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
            endl ≪ sql_strm.str();
        AfxMessageBox(temp_strm.str().c_str());
        temp_strm.str("");
        e→Delete();
      }     /* catch */
      sql_strm.str("");
      continuation_strm.str("");
    }     /* if (curr_str.length() > 0) */
  }     /* else (curr_str.length() > string_length_limit) */
  return 0; }     /* End of Subcategory_Container::f_047I_a definition. */
```

727. Exemplar Production Number (Pica+ 203@ / Pica3 7800).
Exemplar-Produktionsnummer [LDF 2006.08.17.]

Log

[LDF 2006.08.17.] Added this section.

728. Exemplar Production Number (‘0’).

Exemplar-Produktionsnummer (‘0’). [LDF 2006.08.17.]

Log

[LDF 2006.08.17.] Added this function.

[LDF 2006.08.29.] Changed this function to correspond to changes made in the **Exemplar_Production_Numbers** database table: Replaced the column *exemplar_production_number* with the columns *exemplar_production_number_numeri* and *exemplar_production_number_text*. Unlike *exemplar_production_number*, *exemplar_production_number_numeric* can contain `NULL`.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_203_AT_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

729.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_203_AT_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm){
      stringstream temp_strm;
      stringstream sql_strm;
#if 0     /* 1 */
      temp_strm ≪ "In␣‘Subcategory_Container::f_203_AT_0’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
          record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
      Temp_IDs temp_ids;
        /* ***** (5) Error handling: subcategory→field_value contains non-numerals. LDF 2006.08.17. */
```

730.

Log

[LDF 2006.08.29.] Changed this code. Exemplar production numbers can contain non-digits. I've therefore changed the **Exemplar_Production_Numbers** database table accordingly. I 've replaced the column *exemplar_production_number* with the columns *exemplar_production_number_numeric* and *exemplar_production_number_te*
Unlike *exemplar_production_number*, *exemplar_production_number_numeric* can contain `NULL`.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  bool non_digit_switch = false;
  for (unsigned int i = 0; i < subcategory→field_value.size( ); ++i) {
    if (¬isdigit(subcategory→field_value[i])) {
      non_digit_switch = true;
    }    /* if */
  }    /* for */
```

731.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  sql_strm ≪ "delete␣Temp_IDs␣insert␣Temp_IDs␣select␣exemplar_production_number_id␣" ≪
      "from␣Exemplar_Production_Numbers␣";
  if (non_digit_switch)
    sql_strm ≪ "where␣exemplar_production_number_text␣=␣'" ≪ subcategory→field_value ≪
        "'␣and␣exemplar_production_number_numeric␣=␣NULL";
  else sql_strm ≪ "where␣exemplar_production_number_numeric␣=␣" ≪ subcategory→field_value ≪
        "␣and␣exemplar_production_number_text␣=␣'N/A'";
#if 0   /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  try {
    database→ExecuteSQL(sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣'cdb->ExecuteSQL'." ≪ endl ≪ "Error␣code␣=␣" ≪
        e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪ endl ≪
        sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }   /* catch */
  sql_strm.str("");
  temp_ids.Open( );
  if (temp_ids.IsBOF( )) {
#if 0   /* 1 */
    temp_strm ≪ "No␣corresponding␣entry␣in␣'Exemplar_Production_Numbers'." ≪ endl ≪
        "Will␣create␣new␣one.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    sql_strm ≪ "insert␣Exemplar_Production_Numbers␣" ≪ "(exemplar_production_number";
    if (non_digit_switch) sql_strm ≪ "_text";
    else sql_strm ≪ "_numeric";
    sql_strm ≪ ",␣record_id)␣" ≪ "values␣(";
    if (non_digit_switch) sql_strm ≪ "'" ≪ subcategory→field_value ≪ "'";
    else sql_strm ≪ subcategory→field_value;
    sql_strm ≪ ",␣" ≪ record_id ≪ ")";
#if 0   /* 1 */
    temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    try {
      database→ExecuteSQL(sql_strm.str( ).c_str( ));
    }
    catch(CDBException *e)
    {
      temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
          e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
          endl ≪ sql_strm.str( );
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
      e→Delete( );
    }    /* catch */
    sql_strm.str("");
  }    /* if (temp_ids.IsBOF( )) */
  else {
#if 0    /* 1 */
    temp_strm ≪ "A␣corresponding␣entry␣in␣‘Exemplar_Production_Numbers’␣" ≪
        "already␣exists." ≪ endl ≪ "‘temp_ids.m_temp_id’␣==␣" ≪ temp_ids.m_temp_id;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    sql_strm ≪ "if␣not␣exists␣(select␣*␣from␣Exemplar_Production_Numbers␣" ≪
        "where␣record_id␣=␣" ≪ record_id ≪ "␣and␣exemplar_production_number";
    if (non_digit_switch) sql_strm ≪ "_text";
    else sql_strm ≪ "_numeric";
    sql_strm ≪ "␣=␣" ≪ subcategory→field_value ≪ "␣and␣exemplar_production_number";
    if (non_digit_switch) sql_strm ≪ "_numeric␣=␣NULL";
    else sql_strm ≪ "_text␣=␣’N/A’";
    sql_strm ≪ ")␣insert␣Exemplar_Production_Numbers␣" ≪
        "(record_id,␣exemplar_production_number_text,␣" ≪
        "exemplar_production_number_numeric)␣values␣(" ≪ record_id ≪ ",␣";
    if (non_digit_switch) sql_strm ≪ "’" ≪ subcategory→field_value ≪ "’,␣NULL";
    else sql_strm ≪ "’N/A’,␣" ≪ subcategory→field_value;
    sql_strm ≪ subcategory→field_value ≪ ")";
#if 0    /* 1 */
    temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    try {
      database→ExecuteSQL(sql_strm.str( ).c_str( ));
    }
    catch(CDBException *e)
    {
      temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
          e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
          endl ≪ sql_strm.str( );
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
      e→Delete( );
    }    /* catch */
    sql_strm.str("");
  }    /* else */
  if (temp_ids.IsOpen( )) temp_ids.Close( );
#if 0    /* 1 */
```

```
  temp_strm ≪ "Exiting␣'Subcategory_Container::f_203_AT_0':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
      record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  return 0; }    /* End of Subcategory_Container::f_203_AT_0 definition. */
```

732. Call Number (Pica+ 209A / Pica3 7100–7109).
Signatur (Pica+ 209A / Pica3 7100–7109). [LDF 2006.08.17.]

Log

[LDF 2006.08.15.] Added this section.

733. Call Number ('a').
Signatur ('a'). [LDF 2006.08.17.]

Log

[LDF 2006.08.15.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_209A_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

734.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_209A_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "Entering␣'Subcategory_Container::f_209A_a':" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
#if 0     /* 1 */
    temp_strm ≪ "Exiting␣'Subcategory_Container::f_209A_a'.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    return 0;
  }     /* End of Subcategory_Container::f_209A_a definition. */
```

735. Library Number ('b').
Bibliotheksnummer ('b'). [LDF 2006.08.15.]

Log

[LDF 2006.08.15.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_209A_b(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

736.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_209A_b(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_209A_b':" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }     /* End of Subcategory_Container::f_209A_b definition. */
```

737. Special Location ('f').
Sonderstandort ('f'). [LDF 2006.08.15.]

Log

[LDF 2006.08.15.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_209A_f(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

738.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container :: f_209A_f (
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_209A_f’:" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox (temp_strm.str( ).c_str( ));
    temp_strm.str ("");
#endif
    category→database_command_arguments.push_back (make_pair (subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }     /* End of Subcategory_Container :: f_209A_f definition. */
```

739. Library Department (‘j’).
Abteilung der Bibliothek (‘j’). [LDF 2006.08.15.]

Log

[LDF 2006.08.15.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_209A_j (
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

740.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_209A_j(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_209A_j’:" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }    /* End of Subcategory_Container::f_209A_j definition. */
```

741. Access Number (Pica+ 209C / Pica3 8100).
Zugangsnummer (Pica+ 209C / Pica3 8100). [LDF 2006.08.18.]

Log

[LDF 2006.08.18.] Added this section.

---

742. Access Number ('a').
Zugangsnummer ('a'). [LDF 2006.08.18.]

Log

[LDF 2006.08.18.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_209C_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

743.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_209C_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
    stringstream sql_strm;
#if 0      /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_209C_a’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    sql_strm ≪ "if␣not␣exists␣(select␣*␣from␣Access_Numbers␣" ≪ "where␣access_number␣=␣’" ≪
        subcategory→field_value ≪ "’␣and␣record_id␣=␣" ≪ record_id ≪ ")␣" ≪
        "insert␣Access_Numbers␣(access_number,␣record_id)␣" ≪ "values␣(’" ≪
        subcategory→field_value ≪ "’,␣" ≪ record_id ≪ ")";
#if 0      /* 1 */
    temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    try {
      database→ExecuteSQL(sql_strm.str().c_str());
    }
    catch(CDBException *e)
    {
      temp_strm ≪ "Exception␣when␣calling␣‘cdb->ExecuteSQL’." ≪ endl ≪ "Error␣code␣=␣" ≪
          e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪ "SQL␣Code:" ≪
          endl ≪ sql_strm.str();
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
      e→Delete();
    }      /* catch */
    sql_strm.str("");
    return 0;
  }      /* End of Subcategory_Container::f_209C_a definition. */
```

744. Remote Access (Pica+ 209R / Pica3 7133).
Local information regarding remote access to electronic resources (Pica+ 209R / Pica3 7133).
Lokale Angaben zum Zugriff auf elektronische Ressourcen im Fernzugriff (Pica+ 209R / Pica3 7133). [LDF 2006.08.16.]

Log

[LDF 2006.08.16.] Added this section.

745. Format ('0'). [LDF 2006.08.16.]

Log

[LDF 2006.08.16.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_209R_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

746.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_209R_0(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_209R_0':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }    /* End of Subcategory_Container::f_209R_0 definition. */
```

747. URL (Universal Resource Locator) ('a'). [LDF 2006.08.16.]

Log

[LDF 2006.08.16.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_209R_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

748.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_209R_a(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_209R_a':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }    /* End of Subcategory_Container::f_209R_a definition. */
```

749. URN (Universal Resource Name) ('g'). [LDF 2006.08.16.]

Log

[LDF 2006.08.16.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_209R_g(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

750.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_209R_g(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm << "In␣‘Subcategory_Container::f_209R_g’:␣" << endl << "‘record_id’␣==␣" <<
        record_id << endl << "‘subcategory->field_value’␣==␣" << subcategory->field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category->database_command_arguments.push_back(make_pair(subcategory->pica_plus_field_id,
        subcategory->field_value));
    return 0;
  }     /* End of Subcategory_Container::f_209R_g definition. */
```

751. License indicator (‘S’).
Lizenzindikator (‘S’). [LDF 2006.08.15.]

Log

[LDF 2006.08.15.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_209R_S(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

752.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_209R_S(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣‘Subcategory_Container::f_209R_S’:␣" ≪ endl ≪ "‘record_id’␣==␣" ≪
        record_id ≪ endl ≪ "‘subcategory->field_value’␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }     /* End of Subcategory_Container::f_209R_S definition. */
```

753. Internal Remarks (‘x’).
Interne Bemerkungen (‘x’). [LDF 2006.08.16.]

Log

[LDF 2006.08.16.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_209R_x(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

754.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_209R_x(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0     /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_209R_x':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }     /* End of Subcategory_Container::f_209R_x definition. */
```

755. Text for the Web Display ('y').
Text fuer die Web-Anzeige ('y'). [LDF 2006.08.16.]

Log

[LDF 2006.08.16.] Added this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int f_209R_y(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
```

756.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::f_209R_y(
  CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm)
  {
    stringstream temp_strm;
#if 0    /* 1 */
    temp_strm ≪ "In␣'Subcategory_Container::f_209R_y':␣" ≪ endl ≪ "'record_id'␣==␣" ≪
        record_id ≪ endl ≪ "'subcategory->field_value'␣==␣" ≪ subcategory→field_value;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    category→database_command_arguments.push_back(make_pair(subcategory→pica_plus_field_id,
        subcategory→field_value));
    return 0;
  }    /* End of Subcategory_Container::f_209R_y definition. */
```

757. Showing. [LDF 2006.09.19.]

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  int show(stringstream &, string = "");
```

758.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::show(stringstream &local_strm, string prefix)
  {
    if (prefix ≠ "") local_strm ≪ prefix ≪ endl;
    local_strm ≪ "'pica_plus_field_id'␣==␣" ≪ pica_plus_field_id ≪ endl ≪
        "'content_description_german'␣==␣" ≪ content_description_german ≪ endl ≪
        "'content_description_english'␣==␣" ≪ content_description_english ≪ endl ≪
        "'field_value'␣==␣" ≪ field_value ≪ endl;
    return 0;
  }
```

759. Fix string. [LDF Undated.]

Log

[LDF Undated.] Added this function.

[LDF 2006.08.16.] !! BUG FIX: Now setting *out_string* to the empty string at the beginning of this function.

```
⟨ Declare Subcategory_Container functions 572 ⟩ +≡
  static int fix_string(const string &in_string, string &out_string);
```

760.

```
⟨ Define Subcategory_Container functions 573 ⟩ +≡
  int Subcategory_Container::fix_string(const string &in_string, string &out_string)
  {
    char curr_char;
    out_string = "";
    for (string::size_type i = 0; i < in_string.length( ); ++i) {
      curr_char = in_string[i];
      if (curr_char ≡ '\'') out_string += "''";
      else out_string += curr_char;
    }
    return 0;
  }   /* End of Subcategory_Container::fix_string definition. */
```

761. Putting Subcategory_Container together. [LDF Undated.]

762. This is what gets written to `sbctgcnt.h`. [LDF Undated.]

```
⟨ sbctgcnt.h 762 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ "Using" declarations for namespaces 387 ⟩
  ⟨ Declare class Subcategory_Container 569 ⟩
  ⟨ Type definitions 424 ⟩
```

763. This is what gets compiled. [LDF Undated.]

⟨ Include files 13 ⟩
⟨ `sbctgcnt.web` 565 ⟩
⟨ Define **Subcategory_Container** functions 573 ⟩

764. Database_Command (`dbcmmnd.web`). [LDF 2006.09.21.]

Log

[LDF 2006.07.19.] Created `Database_Command.h`.

[LDF 2006.09.21.] Created this file (`dbcmmnd.web`). It contains code formerly in `Database_Command.h` and `Database_Command.cpp`.

⟨ `dbcmmnd.web` 764 ⟩ ≡
static char *id_string*[ ] = `"$Id:␣dbcmmnd.web,v␣1.6␣2006/10/23␣13:36:43␣Administrator␣Exp␣$"`;

This code is cited in sections 6 and 8.
This code is used in section 777.

765. Preprocessor macro calls. [LDF 2006.07.19.]

⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma *once*

766. using declarations for namespaces. [LDF 2006.07.19.]

⟨ **using** declarations for namespaces 191 ⟩ +≡
using namespace std;

767. Include files.

⟨ Include files 13 ⟩ +≡
#include `"stdafx.h"`

768. Forward declarations. [LDF 2006.07.25.]

Log

[LDF 2006.07.25.] Added this section with the forward declarations of **class Category_Container** and **class Subcategory_Container**.

⟨ Forward declarations 399 ⟩ +≡
class Category_Container;
class Subcategory_Container;

769. Database_Command struct declaration.

Log

[LDF 2006.07.19.] Added this **struct** declaration.

[LDF 2006.09.06.] Added **ofstream** &*log_strm* argument to the declaration of *function*.

[LDF 2006.09.06.] Changed the type of **ofstream** &*log_strm* to **Output_Stream_Type** &.

⟨ Declare **struct Database_Command** 769 ⟩ ≡

```
struct Database_Command {
  int(*function)(CDatabase *database,
  long record_id,
  Category_Container *category,
  Subcategory_Container *subcategory,
  Output_Stream_Type &log_strm);
  ⟨ Declare Database_Command functions 771 ⟩
};
```

This code is used in section 776.

770. Functions.

771. Constructor. [LDF 2006.07.19.]

Log

[LDF 2006.07.19.] Added this function.

```
⟨ Declare Database_Command functions 771 ⟩ ≡
  Database_Command(void);
```

See also section 773.

This code is used in section 769.

772.

```
⟨ Define Database_Command functions 772 ⟩ ≡
  Database_Command::Database_Command()
  {
    return;
  }
```

See also section 774.

This code is used in section 777.

773. Destructor. [LDF 2006.07.19.]

Log

[LDF 2006.07.19.] Added this function.

```
⟨ Declare Database_Command functions 771 ⟩ +≡
  ~Database_Command(void);
```

774.

```
⟨ Define Database_Command functions 772 ⟩ +≡
  Database_Command::~Database_Command()
  {
    return;
  }
```

775. Putting Database_Command together.

776. This is what gets written to `dbcmmnd.h`.

⟨ `dbcmmnd.h` 776 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ **using** declarations for namespaces 191 ⟩
  ⟨ Forward declarations 399 ⟩
  ⟨ Declare **struct Database_Command** 769 ⟩

777. This is what gets compiled.

⟨ Include files 13 ⟩
⟨ `dbcmmnd.web` 764 ⟩
⟨ Define **Database_Command** functions 772 ⟩

778. DB_Display (`dbdspl.web`). [LDF 2006.09.21.]

Log

Created this file. It contains code formerly in `dbdspl.h` and `dbdspl.cpp`.

⟨ `dbdspl.web` 778 ⟩ ≡
  static char *id_string*[ ] = `"$Id:␣dbdspl.web,v␣1.8␣2006/12/11␣10:31:36␣Administrator␣Exp␣$"`;

This code is cited in sections 6 and 8.

This code is used in section 996.

779. Preprocessor macro calls. [LDF 2006.07.19.]

⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma *once*

780. using declarations for namespaces. [LDF 2006.09.22.]

⟨ **using** declarations for namespaces 191 ⟩ +≡
  using namespace std;

781. Include files.

⟨ Include files 13 ⟩ +≡
#include `"stdafx.h"`

782. DB_Display class declaration. [LDF 2006.08.23.]
  [LDF 2006.08.23.]

Log

[LDF 2006.08.23.] Added this class declaration with the declarations of the default constructor, the destructor, and the functions *display_single_record*, *display_records*, *open_html_file*, *close_html_file*, and *count_records*.

[LDF 2006.08.24.] Added **const unsigned int** *record_ctr* argument to *display_single_record*.

[LDF 2006.09.06.] In the declaration of *display_records*: Added the optional **CString** *search_command_str* argument with the default `""`.

⟨ Declare **class DB_Display** 782 ⟩ ≡
  class DB_Display {
    ofstream *html_strm*;
  public: ⟨ Declare **DB_Display** functions 784 ⟩
  };

This code is used in section 995.

783. Functions.

784. Constructor. [LDF 2006.08.23.]

Log

[LDF 2006.08.23.] Added this function.

⟨ Declare **DB_Display** functions 784 ⟩ ≡
  DB_Display(**void**);

See also sections 786, 788, 790, 792, 796, and 981.

This code is used in section 782.

785.

⟨ Define **DB_Display** functions 785 ⟩ ≡
  DB_Display :: **DB_Display**(**void**)
  {
    return;
  }

See also sections 787, 789, 791, 793, 794, 797, 798, 799, 800, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, and 993.

This code is used in section 996.

786. Destructor. [LDF 2006.08.23.]

Log

[LDF 2006.08.23.] Added this function.

⟨ Declare **DB_Display** functions 784 ⟩ +≡
  ~DB_Display(**void**);

787.

⟨ Define **DB_Display** functions 785 ⟩ +≡
  DB_Display :: **~DB_Display**(**void**)
  {
    return;
  }

788. Open HTML file.
Open *html_strm* and write header. [LDF 2006.08.23.]

Log

[LDF 2006.08.23.] Added this function.

[LDF 2006.12.11.] Now writing **string** *copyright_hmtl_str* to HTML file.

⟨ Declare **DB_Display** functions 784 ⟩ +≡
  int *open_html_file*(**const char** ∗*html_filename*);

789.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  int DB_Display::open_html_file(const char *html_filename)
  {
    time_t tp;
    time(&tp);
    time_mutex.Lock();
    tm *local_time = localtime(&tp);
    char *datestamp = asctime(local_time);
    time_mutex.Unlock();
    html_strm.open(html_filename);
    html_strm ≪ "<!--␣" ≪ html_filename ≪ "␣-->" ≪ endl ≪ endl ≪
        "<!DOCTYPE␣HTML␣PUBLIC␣" ≪ "\"-//W3C//DTD␣HTML␣4.01␣Transitional//EN\"" ≪
        endl ≪ "\"http://www.w3.org/TR/html4/loose.dtd\">" ≪ endl ≪ "<!␣file:///u|/" ≪
        html_filename ≪ ">" ≪ "<html>\n<head>\n<title>\nRecords␣List\n</title>" ≪
        endl ≪ "</head>\n<body>\n<font␣color=\"black\">" ≪
        "\n<h1␣align=\"center\">\n<a␣name=\"Top\">␣Records␣List" ≪
        "\n</a>\n</h1>\n</font>\n\n" ≪ "<p>\nGenerated␣by␣ZTest,␣" ≪ datestamp ≪
        "</p>\n\n" ≪ copyright_html_str ≪ "<hr␣size=\"2\"␣color=\"black\">\n\n";
#if 0     /* 1 */
    temp_strm ≪ "‘datestamp’␣==␣" ≪ datestamp;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    return 0;
  }
```

790. Close HTML file. [LDF 2006.08.23.]
Write final code to *html_strm* and close it. [LDF 2006.08.23.]

Log

[LDF 2006.08.23.] Added this function.

```
⟨ Declare DB_Display functions 784 ⟩ +≡
  int close_html_file(void);
```

791.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  int DB_Display :: close_html_file(void)
  {
    html_strm ≪ "\n</font>\n</body>\n</html>\n";
    html_strm.close( );
    return 0;
  }
```

792. Counting records. [LDF 2006.08.23.]

Log

[LDF 2006.08.23.] Added this function.

```
⟨ Declare DB_Display functions 784 ⟩ +≡
  int count_records(unsigned int start, unsigned int end, vector⟨long⟩ *record_id_vector = 0);
```

793.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  int DB_Display :: count_records(unsigned int start, unsigned int end, vector⟨long⟩
          *record_id_vector){
#if 0     /* 1 */
      temp_strm ≪ "In␣‘DB_Display::count_records’.";
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
#endif
      int record_ctr = 0;
      stringstream sql_strm;
      stringstream temp_strm;
      Records records;
```

794. Warning: $end > 0 \land end < start$. [LDF 2006.08.23.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (end > 0 ∧ end < start) {
    temp_strm ≪ "WARNING!␣␣In␣‘DB_Display::count_records’:" ≪ endl ≪
        "‘end␣>␣0␣&&␣end␣<␣start’.␣␣This␣isn’t␣allowed." ≪ endl ≪
        "Will␣count␣all␣records␣from␣‘start’.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    end = 0;
  }   /* if (end > 0 ∧ end < start) */   /* **** (4) */
  sql_strm ≪ "select␣*␣from␣Records␣where␣record_id␣>=␣" ≪ start;
  if (end ≥ start) sql_strm ≪ "␣and␣record_id␣<=␣" ≪ end;
#if 0   /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  records.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");   /* **** (4) */
  if (records.IsBOF()) {
#if 0   /* 1 */
    temp_strm ≪ "No␣records␣found.␣␣Returning␣0.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    records.Close();
    if (record_id_vector) record_id_vector→clear();
    return 0;
  }   /* if (records.IsBOF()) */   /* ***** (5) */
  else   /* ¬records.IsBOF() */
  {
    while (¬records.IsEOF()) {
      if (record_id_vector) record_id_vector→push_back(records.m_record_id);
      records.MoveNext();
      ++record_ctr;
    }   /* while */
#if 0   /* 1 */
    temp_strm ≪ "Found␣" ≪ record_ctr;
    if (record_ctr ≡ 1) temp_strm ≪ "␣record.";
    else temp_strm ≪ "␣records.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    records.Close();
    return record_ctr;
  }   /* else (¬records.IsBOF() ) */   /* **** (4) */
  }   /* End of DB_Display::count_records definition. */
```

795. Displaying records.

796. Display single record. [LDF 2006.08.23.]

Log

[LDF 2006.08.23.] Added this function.

[LDF 2006.08.24.] Added **const unsigned int** *record_ctr* argument.

⟨ Declare **DB_Display** functions 784 ⟩ +≡

```
  int display_single_record(const unsigned int record_number, const unsigned int record_ctr);
```

797.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  int DB_Display::display_single_record(const unsigned int record_number, const unsigned int
          record_ctr){
#if 0     /* 1 */
      temp_strm << "In␣‘DB_Display::display_single_record’." << endl <<
          "‘record_number’␣==␣" << record_number;
      AfxMessageBox(temp_strm.str().c_str());
      temp_strm.str("");
#endif
```

798.

Log

[LDF 2006.09.06.] Added the declaration of **Bibliographic_Type_Codes** *bibliographic_type_codes*.

[LDF 2006.09.07.] Added the declaration of **Content_Summaries** *content_summaries*.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  stringstream sql_strm;
  stringstream temp_strm;
  Sources sources;
  Records records;
  Call_Numbers call_numbers;
  Records_Call_Numbers records_call_numbers;
  Access_Numbers access_numbers;
  Exemplar_Production_Numbers exemplar_production_numbers;
  Bibliographic_Type_Codes bibliographic_type_codes;
  Bibliographic_Types bibliographic_types;
  Records_Bibliographic_Types records_bibliographic_types;
  Records_Authors records_authors;
  Authors authors;
  Records_Contributors records_contributors;
  Contributors contributors;
  Records_Main_Titles records_main_titles;
  Main_Titles main_titles;
  Content_Summaries content_summaries;
  Records_Subjects records_subjects;
  Subjects subjects;
  Subject_Types subject_types;
  Permutation_Patterns permutation_patterns;
  Remote_Access remote_access;
  Records_Remote_Access records_remote_access;
  Physical_Descriptions physical_descriptions;
  Records_Physical_Descriptions records_physical_descriptions;
  PICA_Categories pica_categories;
  PICA_Fields pica_fields;
  PICA_Categories_PICA_Fields pica_categories_pica_fields;
  Languages languages;
  Records_Languages records_languages;
  Publishers publishers;
  Records_Publishers records_publishers;
  Database_Providers database_providers;
  Records_Database_Providers records_database_providers;
  vector⟨long⟩ id_vector;
  vector⟨long⟩::iterator id_vector_iter;
```

799. Error handling: *html_strm* is closed. Can't display records. [LDF 2006.08.23.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (¬html_strm.is_open()) {
    temp_strm ≪ "ERROR!␣␣In␣'DB_Display::display_single_record':" ≪ endl ≪
        "'DB_Display::html_strm'␣isn't␣open.␣␣Can't␣display␣records." ≪ endl ≪
        "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    return 1;
  }    /* if (¬html_strm.is_open()) */
```

800.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  unsigned int entry_ctr = 0;

  sql_strm ≪ "select␣*␣from␣Records␣where␣record_id␣=␣" ≪ record_number;
  records.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

801. Try to find entry in **Records** with *record_id* ≡ *record_number*. [LDF 2006.08.23.]

802. Failed to find corresponding entry. This isn't an error, since *display_records* may call this function for increasing values of *record_number* until this happens. [LDF 2006.08.23.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (records.IsBOF()) {
#if 0    /* 1 */
    temp_strm ≪ "In␣'DB_Display::display_single_record':" ≪ endl ≪
        "Failed␣to␣find␣a␣record␣with␣'record_id'␣==␣" ≪ record_number ≪ endl ≪
        "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    records.Close();
    return 1;
  }    /* if (records.IsBOF()) */
```

803. Found entry. [LDF 2006.08.23.]

Note that the conversion of the Transact-SQL *datetime* type to *CTime* in **Records** loses the fractional part of the seconds! [LDF 2006.08.23.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else     /* ¬records.IsBOF( ) */
  {
#if 0     /* 1 */
  temp_strm ≪ "Opening␣'records'␣succeeded." ≪ endl ≪ "'records.m_record_id'␣==␣" ≪
      records.m_record_id;
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  sql_strm ≪ "select␣*␣from␣Sources␣where␣source_id␣=␣" ≪ records.m_source_id;
  sources.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");

  string source_name_str;
  string source_abbrev_str;
  string source_address_str;

  if (sources.IsBOF( )) {
#if 0     /* 1 */
    temp_strm ≪ "Failed␣to␣find␣source␣name␣for␣'source_id'␣==␣" ≪ records.m_source_id;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    source_name_str = "N/A";
    source_abbrev_str = "N/A";
    source_address_str = "N/A";
  }
  else {
    source_name_str = sources.m_source_name;
    source_abbrev_str = sources.m_source_abbrev;
    source_address_str = sources.m_source_address;
  }
  sources.Close( );
```

804.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  html_strm ≪ "<h2␣align=\"center\"><a␣name=\"Record_" ≪ records.m_record_id ≪ "\">␣" ≪
      record_ctr ≪ ".␣Record␣" ≪ records.m_record_id ≪ "</a>\n</h2>\n\n</p>\n" ≪
      "</table>\n<br>\n" ≪ "<table␣border=\"1\">\n<caption␣align=\"left\">" ≪
      "<a␣name=\"Records_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
      "<b>Records␣Table␣" ≪ records.m_record_id ≪ "&nbsp;</b></a></caption>\n" ≪
      "<tr><td>&nbsp;<b>record_id</b>&nbsp;<td>&nbsp;" ≪ records.m_record_id ≪
      "&nbsp;" ≪ "<tr><td>&nbsp;<b>eln_original_entry</b>&nbsp;<td>&nbsp;" ≪
      records.m_eln_original_entry ≪ "&nbsp;" ≪ "<tr><td>&nbsp;<b>eln_most_recent_change\
      </b>&nbsp;<td>&nbsp;" ≪ records.m_eln_most_recent_change ≪ "&nbsp;" ≪
      "<tr><td>&nbsp;<b>eln_status_change</b>&nbsp;<td>&nbsp;" ≪ records.m_eln_status_change ≪
      "&nbsp;" ≪ "<tr><td>&nbsp;<b>source_id</b>&nbsp;<td>&nbsp;" ≪
      records.m_source_id ≪ "&nbsp;" ≪ "<tr><td>&nbsp;<b>source_name</b>␣" ≪
      "(from␣<b>Sources</b>␣table)&nbsp;<td>&nbsp;" ≪ source_name_str ≪
      "&nbsp;" ≪ "<tr><td>&nbsp;<b>source_abbrev</b>␣" ≪
      "(from␣<b>Sources</b>␣table)&nbsp;<td>&nbsp;" ≪ source_abbrev_str ≪
      "&nbsp;" ≪ "<tr><td>&nbsp;<b>source_address</b>␣" ≪
      "(from␣<b>Sources</b>␣table)&nbsp;<td>&nbsp;" ≪ source_address_str ≪
      "&nbsp;" ≪ "<tr><td>&nbsp;<b>identification_number</b>&nbsp;<td>&nbsp;" ≪
      records.m_identification_number ≪ "&nbsp;" ≪ "<tr><td>&nbsp;<b>date_original_entry</b\
      >&nbsp;<td>&nbsp;" ≪ records.m_date_original_entry.Format("%Y-%m-%d␣%H:%M:%S") ≪
      "&nbsp;" ≪ "<tr><td>&nbsp;<b>date_most_recent_change</b>&nbsp;<td>&nbsp;" ≪
      records.m_date_most_recent_change.Format("%Y-%m-%d␣%H:%M:%S") ≪ "&nbsp;" ≪
      "<tr><td>&nbsp;<b>date_status_change</b>&nbsp;<td>&nbsp;" ≪
      records.m_date_status_change.Format("%Y-%m-%d␣%H:%M:%S") ≪ "&nbsp;";
```

805.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  html_strm ≪ "<tr><td>&nbsp;<b>year_appearance_begin</b>&nbsp;<td>&nbsp;";
  if (records.IsFieldNull(&records.m_year_appearance_begin)) html_strm ≪ "NULL&nbsp;";
  else html_strm ≪ records.m_year_appearance_begin ≪ "&nbsp;";
  html_strm ≪ "<tr><td>&nbsp;<b>year_appearance_end</b>&nbsp;<td>&nbsp;";
  if (records.IsFieldNull(&records.m_year_appearance_end)) html_strm ≪ "NULL&nbsp;";
  else html_strm ≪ records.m_year_appearance_end ≪ "&nbsp;";
  html_strm ≪ "<tr><td>&nbsp;<b>year_appearance_rak_wb</b>&nbsp;<td>&nbsp;";
  if (records.IsFieldNull(&records.m_year_appearance_rak_wb)) html_strm ≪ "NULL&nbsp;";
  else html_strm ≪ records.m_year_appearance_rak_wb ≪ "&nbsp;";
  html_strm ≪ "<tr><td>&nbsp;<b>year_appearance_original</b>&nbsp;<td>&nbsp;";
  if (records.IsFieldNull(&records.m_year_appearance_original)) html_strm ≪ "NULL&nbsp;";
  else html_strm ≪ records.m_year_appearance_original ≪ "&nbsp;";
  html_strm ≪ "\n</table>\n<br>\n";
#if 0    /* 1 */
#endif
  records.Close(); }     /* else (¬records.IsBOF()) */
```

806. Records_Call_Numbers [LDF 2006.08.24.]

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  sql_strm ≪ "select␣*␣from␣Records_Call_Numbers␣where␣record_id␣=␣" ≪ record_number;
  records_call_numbers.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

807. No entries found. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (records_call_numbers.IsBOF()) {
#if 0    /* 1 */
    temp_strm ≪ "No␣entries␣in␣'Records_Call_Numbers'␣with␣" ≪ "'record_id'␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Call_Numbers_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Call_Numbers␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }    /* if (records_call_numbers.IsBOF()) */
```

808. Found entries. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* Found entries in Records_Call_Numbers. */
  { html_strm ≪ "<table␣border=\"1\"><caption␣align=\"left\">" ≪
      "<a␣name=\"Call_Numbers_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
      "<b>Call_Numbers␣Table␣" ≪ records.m_record_id ≪ "&nbsp;</b></a></caption>\n" ≪
      "<tr><th>&nbsp;<b>call_number_id</b>&nbsp;<th>&nbsp;" ≪
      "<b>call_number</b>&nbsp;<th>&nbsp;<b>library_number</b>&nbsp;" ≪
      "<th>&nbsp;<b>library_department</b>&nbsp;" ≪
      "<th>&nbsp;<b>special_location</b>&nbsp;";
  entry_ctr = 0;
  id_vector.clear();
```

809.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬records_call_numbers.IsEOF()) {
    id_vector.push_back(records_call_numbers.m_call_number_id);
    records_call_numbers.MoveNext();
    ++entry_ctr;
  }    /* while */
```

810.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0    /* 1 */
  temp_strm << "‘id_vector’:" << endl;
#endif
  id_vector_iter = id_vector.begin( );
  sql_strm << "select␣*␣from␣Call_Numbers␣where␣call_number_id␣=␣" << *id_vector_iter++;
  for ( ; id_vector_iter ≠ id_vector.end( ); ++id_vector_iter) {
#if 0    /* 1 */
    temp_strm << *id_vector_iter << endl;
#endif
    sql_strm << "␣or␣call_number_id␣=␣" << *id_vector_iter;
  }    /* for */
#if 0    /* 1 */
  temp_strm << "SQL␣Code:\n" << sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  call_numbers.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

811.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (call_numbers.IsBOF( )) {
    temp_strm << "WARNING!␣␣In␣‘DB_Display::display_single_record’:" << endl <<
        "No␣corresponding␣entries␣in␣the␣‘Call_Numbers’␣table." << endl << "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
  }    /* if (call_numbers.IsBOF( )) */
```

812.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* ¬call_numbers.IsBOF( ) */
  {
```

813.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬call_numbers.IsEOF( )) {
    html_strm << "\n<tr><td>" << call_numbers.m_call_number_id << "<td>" <<
        call_numbers.m_call_number << "&nbsp;" << "<td>" << call_numbers.m_library_number <<
        "&nbsp;" << "<td>" << call_numbers.m_library_department << "&nbsp;" << "<td>" <<
        call_numbers.m_special_location << "&nbsp;";
    call_numbers.MoveNext( );
  }    /* while */
```

814.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else (¬call_numbers.IsBOF( )) */
```

815.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<br>\n</table>\n<br>\n";
  call_numbers.Close( ); }     /* else Found entries in Records_Call_Numbers. */
  records_call_numbers.Close( );
```

816. Access_Numbers [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣Access_Numbers␣where␣record_id␣=␣" ≪ record_number;
  access_numbers.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

817. No entries found. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (access_numbers.IsBOF( )) {
#if 0     /* 1 */
    temp_strm ≪ "No␣entries␣in␣'Access_Numbers'␣with␣" ≪ "'record_id'␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Access_Numbers_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Access␣Numbers␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a>\n</p>\n\n";
  }     /* if (access_numbers.IsBOF( )) */
```

818. Found entries. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else     /* ¬access_numbers.IsBOF( ) */
  { html_strm ≪ "<table␣border=\"1\"><caption␣align=\"left\">" ≪
      "<a␣name=\"Access_Numbers_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
      "<b>Access_Numbers␣Table␣" ≪ records.m_record_id ≪ "&nbsp;</b></a></caption>\n" ≪
      "<tr><th>&nbsp;<b>access_number_id</b>&nbsp;<th>&nbsp;" ≪
      "<b>access_number</b>&nbsp;<th>&nbsp;<b>record_id</b>&nbsp;";
  entry_ctr = 0;
  while (¬access_numbers.IsEOF( )) {
    html_strm ≪ "\n<tr><td>" ≪ access_numbers.m_access_number_id ≪ "<td>" ≪
        access_numbers.m_access_number ≪ "&nbsp;" ≪ "<td>" ≪ access_numbers.m_record_id ≪
        "&nbsp;";
    access_numbers.MoveNext( );
    ++entry_ctr;
  }     /* while */
```

819.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<br>\n</table><br>\n"; }     /* else (¬access_numbers.IsBOF( )) */
```

820.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  access_numbers.Close( );
```

821. Records_Remote_Access Vertical table, separate database table for *record_id*. [LDF 2006.08.25.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣Records_Remote_Access␣where␣record_id␣=␣" ≪ record_number;
  records_remote_access.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

822. No entries found. [LDF 2006.08.25.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (records_remote_access.IsBOF( )) {
#if 0     /* 1 */
    temp_strm ≪ "No␣entries␣in␣‘Records_Remote_Access’␣with␣" ≪ "‘record_id’␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Remote_Access_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Remote_Access␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }     /* if (records_remote_access.IsBOF( )) */
```

823. Found entries. [LDF 2006.08.25.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else     /* Found entries in Records_Remote_Access. */
  { entry_ctr = 0;
  id_vector.clear( );
```

824.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬records_remote_access.IsEOF( )) {
    id_vector.push_back(records_remote_access.m_remote_access_id);
    records_remote_access.MoveNext( );
    ++entry_ctr;
  }     /* while */
```

825.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0    /* 1 */
  temp_strm ≪ "‘id_vector’:" ≪ endl;
#endif
  id_vector_iter = id_vector.begin( );
  sql_strm ≪ "select␣*␣from␣Remote_Access␣where␣remote_access_id␣=␣" ≪ *id_vector_iter++;
  for ( ; id_vector_iter ≠ id_vector.end( ); ++id_vector_iter) {
#if 0    /* 1 */
    temp_strm ≪ *id_vector_iter ≪ endl;
#endif
    sql_strm ≪ "␣or␣remote_access_id␣=␣" ≪ *id_vector_iter;
  }    /* for */
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  remote_access.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

826.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (remote_access.IsBOF( )) {
    temp_strm ≪ "WARNING!␣␣In␣‘DB_Display::display_single_record’:" ≪ endl ≪
        "No␣corresponding␣entries␣in␣the␣‘Remote_Access’␣table." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
  }    /* if (remote_access.IsBOF( )) */
```

827.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* ¬remote_access.IsBOF( ) */
  {
```

828.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬remote_access.IsEOF()) {
    html_strm ≪ "<table␣border=\"1\">\n" ≪ "<caption␣align=\"left\">" ≪
        "<a␣name=\"Remote_Access_Table_" ≪ records.m_record_id ≪
        "\">␣" ≪ "<b>Remote␣Access␣Table␣" ≪ records.m_record_id ≪
        "&nbsp;</b></a></caption>\n" ≪ "<tr><td>&nbsp;<b>remote_access_id</b>&nbsp;" ≪
        "<td>&nbsp;" ≪ remote_access.m_remote_access_id ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>license_indicator␣␣␣</b>&nbsp;" ≪ "<td>&nbsp;" ≪
        remote_access.m_license_indicator ≪ "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>fo\
        rmat_type</b>&nbsp;" ≪ "<td>&nbsp;" ≪ remote_access.m_format_type ≪
        "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>web_display_text</b>&nbsp;" ≪
        "<td>&nbsp;" ≪ remote_access.m_web_display_text ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>URL</b>&nbsp;" ≪ "<td>&nbsp;" ≪ remote_access.m_URL ≪
        "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>URN</b>&nbsp;" ≪ "<td>&nbsp;" ≪
        remote_access.m_URN ≪ "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>internal_remarks</b>&nbsp;" ≪
        "<td>&nbsp;" ≪ remote_access.m_internal_remarks ≪ "&nbsp;\n";
    html_strm ≪ "</table>\n<br>\n";
    remote_access.MoveNext();
  }    /* while */
```

829.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else (¬remote_access.IsBOF()) */
```

830.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  remote_access.Close();
```

831.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else Found entries in Remote_Access_Records. */
  records_remote_access.Close();
```

832. Exemplar_Production_Numbers [LDF 2006.08.24.]

Log

[LDF 2006.08.29.] Changed this code to correspond to changes made in the **Exemplar_Production_Numbers** database table: Replaced the column *exemplar_production_number* with the columns *exemplar_production_number_numeri* and *exemplar_production_number_text*. Unlike *exemplar_production_number*, *exemplar_production_number_numeric* can contain `NULL`.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣Exemplar_Production_Numbers␣where␣record_id␣=␣" ≪ record_number;
  exemplar_production_numbers.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

833. No entries found. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (exemplar_production_numbers.IsBOF( )) {
#if 0    /* 1 */
    temp_strm ≪ "No␣entries␣in␣‘Exemplar_Production_Numbers’␣with␣" ≪ "‘record_id’␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Exemplar_Production_Numbers_Table_" ≪ records.m_record_id ≪
        "\">␣" ≪ "<b>No␣Exemplar_Production␣Numbers␣for␣Record␣" ≪ records.m_record_id ≪
        "</b></a></p>\n\n";
  }    /* if (exemplar_production_numbers.IsBOF( )) */
```

834. Found entries. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* ¬exemplar_production_numbers.IsBOF( ) */
  { html_strm ≪ "<table␣border=\"1\"><caption␣align=\"left\">" ≪
      "<a␣name=\"Exemplar_Production_Numbers_Table_" ≪ records.m_record_id ≪
      "\">␣" ≪ "<b>Exemplar_Production_Numbers␣Table␣" ≪
      records.m_record_id ≪ "&nbsp;</b></a></caption>\n" ≪
      "<tr><th>&nbsp;<b>exemplar_production_number_id</b>&nbsp;" ≪
      "<th>&nbsp;<b>exemplar_production_number_numeric</b>&nbsp;" ≪
      "<th>&nbsp;<b>exemplar_production_number_text</b>&nbsp;" ≪
      "<th>&nbsp;<b>record_id</b>&nbsp;";
  entry_ctr = 0;
  while (¬exemplar_production_numbers.IsEOF( )) {
    html_strm ≪ "\n<tr><td>" ≪ exemplar_production_numbers.m_exemplar_production_number_id ≪
        "<td>";
    if (exemplar_production_numbers.IsFieldNull(&exemplar_production_numbers.m_exemplar_production_number_nume
      html_strm ≪ "NULL&nbsp;";
    else html_strm ≪ exemplar_production_numbers.m_exemplar_production_number_numeric;
    html_strm ≪ "&nbsp;" ≪ "<td>" ≪
        exemplar_production_numbers.m_exemplar_production_number_text ≪ "&nbsp;" ≪
        "<td>" ≪ exemplar_production_numbers.m_record_id ≪ "&nbsp;";
    exemplar_production_numbers.MoveNext( );
    ++entry_ctr;
  }    /* while */
```

835.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<br>\n</table><br>\n"; }    /* else (¬exemplar_production_numbers.IsBOF( )) */
```

836.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  exemplar_production_numbers.Close( );
```

837. Records_Bibliographic_Types Vertical table, separate database table for *record_id*. [LDF 2006.08.24.] ■

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  sql_strm ≪ "select␣*␣from␣Records_Bibliographic_Types␣where␣record_id␣=␣" ≪ record_number;
  records_bibliographic_types.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

838. No entries found. [LDF 2006.08.24.]

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  if (records_bibliographic_types.IsBOF( )) {
#if 0    /* 1 */
    temp_strm ≪ "No␣entries␣in␣‘Records_Bibliographic_Types’␣with␣" ≪ "‘record_id’␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Bibliographic_Types_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Bibliographic_Types␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }    /* if (records_bibliographic_types.IsBOF( )) */
```

839. Found entries. [LDF 2006.08.24.]

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  else    /* Found entries in Records_Bibliographic_Types. */
  { entry_ctr = 0;
  id_vector.clear( );
```

840.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  while (¬records_bibliographic_types.IsEOF( )) {
    id_vector.push_back(records_bibliographic_types.m_bibliographic_type_id);
    records_bibliographic_types.MoveNext( );
    ++entry_ctr;
  }    /* while */
```

841.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0    /* 1 */
  temp_strm << "‘id_vector’:" << endl;
#endif
  id_vector_iter = id_vector.begin();
  sql_strm << "select␣*␣from␣Bibliographic_Types␣where␣bibliographic_type_id␣=␣" <<
      *id_vector_iter++;
  for ( ; id_vector_iter ≠ id_vector.end(); ++id_vector_iter) {
#if 0    /* 1 */
    temp_strm << *id_vector_iter << endl;
#endif
    sql_strm << "␣or␣bibliographic_type_id␣=␣" << *id_vector_iter;
  }    /* for */
#if 0    /* 1 */
  temp_strm << "SQL␣Code:\n" << sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  bibliographic_types.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

842.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (bibliographic_types.IsBOF()) {
    temp_strm << "WARNING!␣␣In␣‘DB_Display::display_single_record’:" << endl <<
        "No␣corresponding␣entries␣in␣the␣‘Bibliographic_Types’␣table." << endl <<
        "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
  }    /* if (bibliographic_types.IsBOF()) */
```

843.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* ¬bibliographic_types.IsBOF() */
  {
```

844.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬bibliographic_types.IsEOF()) { html_strm << "<table␣border=\"1\">\n" <<
      "<caption␣align=\"left\">" << "<a␣name=\"Bibliographic_Types_Table_" <<
      records.m_record_id << "\">␣" << "<b>Bibliographic␣Types␣Table␣" << records.m_record_id <<
      "&nbsp;</b></a></caption>\n" << "<tr><td>&nbsp;<b>bibliographic_type_id</b>&nbsp;" <<
      "<td>&nbsp;" << bibliographic_types.m_bibliographic_type_id << "&nbsp;\n"
```

845. *physical_form*. [LDF 2006.09.06.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  << "<tr><td>&nbsp;<b>physical_form</b>&nbsp;" << "<td>&nbsp;" <<
      bibliographic_types.m_physical_form << "&nbsp;\n";
```

846. *physical_form*. Data from **Bibliographic_Type_Codes** table. [LDF 2006.09.06.]

Log

[LDF 2006.09.06.] Added this section.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣" ≪ "Bibliographic_Type_Codes␣where␣physical_form_code␣=␣'" ≪
      bibliographic_types.m_physical_form ≪ "'";
  bibliographic_type_codes.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

847. Material Name English [LDF 2006.09.06.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<tr><td>&nbsp;<b>physical_form_material_name_english</b>␣" ≪
      "(from␣<b>Bibliographic_Type_Codes</b>␣table)&nbsp;" ≪ "<td>&nbsp;";
```

848.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (bibliographic_type_codes.IsBOF( )) {
    html_strm ≪ "INVALID␣PHYSICAL_FORM␣CODE";
  }
```

849.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else
    if (bibliographic_type_codes.IsFieldNull(
          &bibliographic_type_codes.m_physical_form_material_name_english)) {
      html_strm ≪ "NULL";
    }
```

850.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else {
    html_strm ≪ bibliographic_type_codes.m_physical_form_material_name_english;
  }
```

851.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "&nbsp;\n";
```

852. Material Name German. [LDF 2006.09.06.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<tr><td>&nbsp;<b>physical_form_material_name_german</b>␣" ≪
      "(from␣<b>Bibliographic_Type_Codes</b>␣table)&nbsp;" ≪ "<td>&nbsp;";
  if (bibliographic_type_codes.IsBOF()) {
    html_strm ≪ "INVALID␣PHYSICAL_FORM␣CODE";
  }
  else if (bibliographic_type_codes.IsFieldNull(
        &bibliographic_type_codes.m_physical_form_material_name_german)) {
    html_strm ≪ "NULL";
  }
  else {
    html_strm ≪ bibliographic_type_codes.m_physical_form_material_name_german;
  }
  html_strm ≪ "&nbsp;\n";
  bibliographic_type_codes.Close();
```

853. *bibliographic_representation*. [LDF 2006.09.06.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<tr><td>&nbsp;<b>bibliographic_representation</b>&nbsp;" ≪ "<td>&nbsp;" ≪
      bibliographic_types.m_bibliographic_representation ≪ "&nbsp;\n";
```

854. *bibliographic_representation*. Data from **Bibliographic_Type_Codes** table. [LDF 2006.09.06.]

Log

[LDF 2006.09.06.] Added this section.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣" ≪ "Bibliographic_Type_Codes␣where␣" ≪
      "bibliographic_representation_code␣=␣'" ≪ bibliographic_types.m_bibliographic_representation ≪
      "'";
  bibliographic_type_codes.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

855. Description English [LDF 2006.09.06.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<tr><td>&nbsp;<b>" ≪ "bibliographic_representation_descriptio\
     n_english</b>␣" ≪ "(from␣<b>Bibliographic_Type_Codes</b>␣table)&nbsp;" ≪
     "<td>&nbsp;";
  if (bibliographic_type_codes.IsBOF()) {
    html_strm ≪ "INVALID␣BIBLIOGRAPHIC_REPRESENTATION␣CODE";
  }
  else if (bibliographic_type_codes.IsFieldNull(
        &bibliographic_type_codes.m_bibliographic_representation_description_english)) {
    html_strm ≪ "NULL";
  }
  else {
    html_strm ≪ bibliographic_type_codes.m_bibliographic_representation_description_english;
  }
  html_strm ≪ "&nbsp;\n";
```

856. Description German. [LDF 2006.09.06.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<tr><td>&nbsp;<b>bibliographic_representation_description_german</b>␣" ≪
      "(from␣<b>Bibliographic_Type_Codes</b>␣table)&nbsp;" ≪ "<td>&nbsp;";
  if (bibliographic_type_codes.IsBOF()) {
    html_strm ≪ "INVALID␣BIBLIOGRAPHIC_REPRESENTATION␣CODE";
  }
  else if (bibliographic_type_codes.IsFieldNull(
        &bibliographic_type_codes.m_bibliographic_representation_description_german)) {
    html_strm ≪ "NULL";
  }
  else {
    html_strm ≪ bibliographic_type_codes.m_bibliographic_representation_description_german;
  }
  html_strm ≪ "&nbsp;\n";
  bibliographic_type_codes.Close();
```

857. *description_status* [LDF 2006.09.06.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<tr><td>&nbsp;<b>description_status</b>&nbsp;" ≪ "<td>&nbsp;" ≪
      bibliographic_types.m_description_status ≪ "&nbsp;\n";
```

858. *description_status*. Data from **Bibliographic_Type_Codes** table. [LDF 2006.09.06.]

Log

[LDF 2006.09.06.] Added this section.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣" ≪ "Bibliographic_Type_Codes␣where␣descript\
      ion_status_code␣=␣'" ≪ bibliographic_types.m_description_status ≪ "'";
  bibliographic_type_codes.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

859. Description English [LDF 2006.09.06.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<tr><td>&nbsp;<b>description_status_description_english</b>␣" ≪
      "(from␣<b>Bibliographic_Type_Codes</b>␣table)&nbsp;" ≪ "<td>&nbsp;";
  if (bibliographic_type_codes.IsBOF()) {
    html_strm ≪ "INVALID␣DESCRIPTION_STATUS␣CODE";
  }
  else if (bibliographic_type_codes.IsFieldNull(
        &bibliographic_type_codes.m_description_status_description_english)) {
    html_strm ≪ "NULL";
  }
  else {
    html_strm ≪ bibliographic_type_codes.m_description_status_description_english;
  }
  html_strm ≪ "&nbsp;\n";
```

860. Description German. [LDF 2006.09.06.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<tr><td>&nbsp;<b>description_status_description_german</b>␣" ≪
      "(from␣<b>Bibliographic_Type_Codes</b>␣table)&nbsp;" ≪ "<td>&nbsp;";
  if (bibliographic_type_codes.IsBOF()) {
    html_strm ≪ "INVALID␣DESCRIPTION_STATUS␣CODE";
  }
  else if (bibliographic_type_codes.IsFieldNull(
        &bibliographic_type_codes.m_description_status_description_german)) {
    html_strm ≪ "NULL";
  }
  else {
    html_strm ≪ bibliographic_type_codes.m_description_status_description_german;
  }
  html_strm ≪ "&nbsp;\n";
  bibliographic_type_codes.Close();
  html_strm ≪ "<tr><td>&nbsp;<b>miscellaneous</b>&nbsp;" ≪ "<td>&nbsp;";
  if (bibliographic_types.IsFieldNull(
        &bibliographic_types.m_miscellaneous)) html_strm ≪ "NULL&nbsp;";
  else html_strm ≪ bibliographic_types.m_miscellaneous ≪ "&nbsp;\n";
  html_strm ≪ "<tr><td>&nbsp;<b>bibliographic_representation_refinement</b>&nbsp;" ≪
      "<td>&nbsp;";
  if (bibliographic_types.IsFieldNull(
        &bibliographic_types.m_bibliographic_representation_refinement)) html_strm ≪ "NULL&nbsp;";
  else html_strm ≪ bibliographic_types.m_bibliographic_representation_refinement ≪ "&nbsp;\n";
  html_strm ≪ "<tr><td>&nbsp;<b>transliteration_code</b>&nbsp;" ≪ "<td>&nbsp;";
  if (bibliographic_types.IsFieldNull(
        &bibliographic_types.m_transliteration_code)) html_strm ≪ "NULL&nbsp;";
  else html_strm ≪ bibliographic_types.m_transliteration_code ≪ "&nbsp;\n";
  html_strm ≪ "</table>\n<br>\n";
  bibliographic_types.MoveNext(); }    /* while */
```

861.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else (¬bibliographic_types.IsBOF()) */
```

862.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  bibliographic_types.Close(); }    /* else Found entries in Bibliographic_Types_Records. */
  records_bibliographic_types.Close();
```

863. Records_Authors [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣Records_Authors␣where␣record_id␣=␣" ≪ record_number;
  records_authors.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

864. No entries found. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (records_authors.IsBOF()) {
#if 0     /* 1 */
    temp_strm ≪ "No␣entries␣in␣'Records_Authors'␣with␣" ≪ "'record_id'␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Authors_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Authors␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }     /* if (records_authors.IsBOF()) */
```

865. Found entries. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else     /* Found entries in Records_Authors. */
  { html_strm ≪ "<table␣border=\"1\"><caption␣align=\"left\">" ≪ "<a␣name=\"Authors_T\
      able_" ≪ records.m_record_id ≪ "\">␣" ≪ "<b>Authors␣Table␣" ≪ records.m_record_id ≪
      "&nbsp;</b></a></caption>\n" ≪ "<tr><th>&nbsp;<b>author_id</b>&nbsp;" ≪
      "<th>&nbsp;<b>surname</b>&nbsp;<th>&nbsp;<b>given_name</b>" ≪ "&nbsp;" ≪
      "<th>&nbsp;<b>prefix</b>&nbsp;<th>&nbsp;" ≪ "<b>id_number_ppn</b>&nbsp;";
  entry_ctr = 0;
  id_vector.clear();
```

866.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬records_authors.IsEOF()) {
    id_vector.push_back(records_authors.m_author_id);
    records_authors.MoveNext();
    ++entry_ctr;
  }     /* while */
```

867.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0   /* 1 */
  temp_strm ≪ "‘id_vector’:" ≪ endl;
#endif
  id_vector_iter = id_vector.begin( );
  sql_strm ≪ "select␣*␣from␣Authors␣where␣author_id␣=␣" ≪ *id_vector_iter++;
  for ( ; id_vector_iter ≠ id_vector.end( ); ++id_vector_iter) {
#if 0   /* 1 */
    temp_strm ≪ *id_vector_iter ≪ endl;
#endif
    sql_strm ≪ "␣or␣author_id␣=␣" ≪ *id_vector_iter;
  }   /* for */
#if 0   /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  authors.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

868.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (authors.IsBOF( )) {
    temp_strm ≪ "WARNING!␣␣In␣‘DB_Display::display_single_record’:" ≪ endl ≪
        "No␣corresponding␣entries␣in␣the␣‘Authors’␣table." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
  }   /* if (authors.IsBOF( )) */
```

869.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else   /* ¬authors.IsBOF( ) */
  {
```

870.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬authors.IsEOF( )) {
    html_strm ≪ "\n<tr><td>&nbsp;" ≪ authors.m_author_id ≪ "&nbsp;" ≪ "<td>&nbsp;" ≪
        authors.m_surname ≪ "&nbsp;" ≪ "<td>&nbsp;" ≪ authors.m_given_name ≪ "&nbsp;" ≪
        "<td>&nbsp;" ≪ authors.m_prefix ≪ "&nbsp;" ≪ "<td>&nbsp;" ≪ authors.m_id_number_ppn ≪
        "&nbsp;";
    authors.MoveNext( );
  }   /* while */
```

871.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }   /* else (¬authors.IsBOF( )) */
```

872.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<br>\n</table>\n<br>\n";
  authors.Close( );
```

873.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else Found entries in Records_Authors. */
  records_authors.Close( );
```

874. Records_Contributors Horizontal table, separate database table for *record_id*. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣Records_Contributors␣where␣record_id␣=␣" ≪ record_number;
  records_contributors.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

875. No entries found. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (records_contributors.IsBOF( )) {
#if 0    /* 1 */
    temp_strm ≪ "No␣entries␣in␣‘Records_Contributors’␣with␣" ≪ "‘record_id’␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Contributors_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Contributors␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }    /* if (records_contributors.IsBOF( )) */
```

876. Found entries. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* Found entries in Records_Contributors. */
  { html_strm ≪ "<table␣border=\"1\"><caption␣align=\"left\">" ≪
      "<a␣name=\"Contributors_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
      "<b>Contributors␣Table␣" ≪ records.m_record_id ≪ "&nbsp;</b></a></caption>\n" ≪
      "<tr><th>&nbsp;<b>contributor_id</b>&nbsp;<th>&nbsp;" ≪
      "<b>surname</b>&nbsp;<th>&nbsp;<b>given_name</b>&nbsp;" ≪
      "<th>&nbsp;<b>prefix</b>&nbsp;" ≪ "<th>&nbsp;<b>id_number_ppn</b>&nbsp;";
  entry_ctr = 0;
  id_vector.clear( );
```

877.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬records_contributors.IsEOF( )) {
    id_vector.push_back(records_contributors.m_contributor_id);
    records_contributors.MoveNext( );
    ++entry_ctr;
  }    /* while */
```

878.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0   /* 1 */
  temp_strm ≪ "'id_vector':" ≪ endl;
#endif
  id_vector_iter = id_vector.begin( );
  sql_strm ≪ "select␣*␣from␣Contributors␣where␣contributor_id␣=␣" ≪ *id_vector_iter++;
  for ( ; id_vector_iter ≠ id_vector.end( ); ++id_vector_iter) {
#if 0   /* 1 */
    temp_strm ≪ *id_vector_iter ≪ endl;
#endif
    sql_strm ≪ "␣or␣contributor_id␣=␣" ≪ *id_vector_iter;
  }   /* for */
#if 0   /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  contributors.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

879.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (contributors.IsBOF( )) {
    temp_strm ≪ "WARNING!␣␣In␣'DB_Display::display_single_record':" ≪ endl ≪
        "No␣corresponding␣entries␣in␣the␣'Contributors'␣table." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
  }   /* if (contributors.IsBOF( )) */
```

880.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* ¬contributors.IsBOF( ) */
  {
```

881.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬contributors.IsEOF( )) {
    html_strm ≪ "\n<tr><td>&nbsp;" ≪ contributors.m_contributor_id ≪ "&nbsp;" ≪ "<td>&nbsp;" ≪
        contributors.m_surname ≪ "&nbsp;" ≪ "<td>&nbsp;" ≪ contributors.m_given_name ≪
        "&nbsp;" ≪ "<td>&nbsp;" ≪ contributors.m_prefix ≪ "&nbsp;" ≪ "<td>&nbsp;" ≪
        contributors.m_id_number_ppn ≪ "&nbsp;";
    contributors.MoveNext( );
  }   /* while */
```

882.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }   /* else (¬contributors.IsBOF( )) */
```

883.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<br>\n</table>\n<br>\n";
  contributors.Close( );
```

884.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else Found entries in Records_Contributors. */
  records_contributors.Close( );
```

885. Records_Main_Titles [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣Records_Main_Titles␣where␣record_id␣=␣" ≪ record_number;
  records_main_titles.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

886. No entries found. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (records_main_titles.IsBOF( )) {
#if 0    /* 1 */
    temp_strm ≪ "No␣entries␣in␣'Records_Main_Titles'␣with␣" ≪ "'record_id'␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Main_Titles_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Main␣Titles␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }    /* if (records_main_titles.IsBOF( )) */
```

887. Found entries. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* Found entries in Records_Main_Titles. */
  { entry_ctr = 0;
  id_vector.clear( );
```

888.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬records_main_titles.IsEOF( )) {
    id_vector.push_back(records_main_titles.m_main_title_id);
    records_main_titles.MoveNext( );
    ++entry_ctr;
  }    /* while */
```

889.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0     /* 1 */
  temp_strm << "‘id_vector’:" << endl;
#endif
  id_vector_iter = id_vector.begin();
  sql_strm << "select␣*␣from␣Main_Titles␣where␣main_title_id␣=␣" << *id_vector_iter++;
  for ( ; id_vector_iter ≠ id_vector.end(); ++id_vector_iter) {
#if 0     /* 1 */
    temp_strm << *id_vector_iter << endl;
#endif
    sql_strm << "␣or␣main_title_id␣=␣" << *id_vector_iter;
  }     /* for */
#if 0     /* 1 */
  temp_strm << "SQL␣Code:\n" << sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  main_titles.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

890.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (main_titles.IsBOF()) {
    temp_strm << "WARNING!␣␣In␣‘DB_Display::display_single_record’:" << endl <<
        "No␣corresponding␣entries␣in␣the␣‘Main_Titles’␣table." << endl << "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
  }     /* if (main_titles.IsBOF()) */
```

891.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else     /* ¬main_titles.IsBOF() */
  {
```

892.

Log

[LDF 2006.09.05.] Changed **Main_Titles** :: **long** *m_continuation* to *m_continuation_main_canonical_title*. I need another column for continuations to *additions_main*.

[LDF 2006.09.05.] Added code for **Main_Titles** :: *m_continuation_additions_main*.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<p>\n<a␣name=\"Main_Titles_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
      "<b>Main_Titles␣Table␣" ≪ records.m_record_id ≪ "</b></a>\n</p>\n\n";
  while (¬main_titles.IsEOF()) {
    html_strm ≪ "<table␣border=\"1\">\n" ≪ "<tr><td>&nbsp;<b>main_title_id</b>&nbsp;" ≪
        "<td>&nbsp;" ≪ main_titles.m_main_title_id ≪ "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>st\
        andard_text</b>&nbsp;" ≪ "<td>&nbsp;" ≪ main_titles.m_standard_text ≪
        "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>main_canonical_title</b>&nbsp;\n" ≪
        "<td>&nbsp;" ≪ main_titles.m_main_canonical_title ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>continuation_main_canonical_title</b>&nbsp;\n" ≪
        "<td>&nbsp;" ≪ main_titles.m_continuation_main_canonical_title ≪
        "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>additions_main</b>&nbsp;\n" ≪
        "<td>&nbsp;" ≪ main_titles.m_additions_main ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>continuation_additions_main</b>&nbsp;\n" ≪
        "<td>&nbsp;" ≪ main_titles.m_continuation_additions_main ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>additional_creator_main</b>&nbsp;\n" ≪
        "<td>&nbsp;" ≪ main_titles.m_additional_creator_main ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>parallel_canonical_title</b>&nbsp;\n" ≪
        "<td>&nbsp;" ≪ main_titles.m_parallel_canonical_title ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>additions_parallel</b>&nbsp;\n" ≪
        "<td>&nbsp;" ≪ main_titles.m_additions_parallel ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>additional_creator_parallel</b>&nbsp;\n" ≪
        "<td>&nbsp;" ≪ main_titles.m_additional_creator_parallel ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>authorship</b>&nbsp;\n" ≪ "<td>&nbsp;" ≪ main_titles.m_authorship ≪
        "&nbsp;\n</table>\n<br>\n";
    main_titles.MoveNext();
  }    /* while */
```

893.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else (¬main_titles.IsBOF()) */
```

894.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  main_titles.Close();
```

895.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else Found entries in Main_Titles_Records. */
  records_main_titles.Close();
```

896. Content_Summaries [LDF 2006.09.07.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣Content_Summaries␣where␣record_id␣=␣" ≪ record_number;
  try {
    content_summaries.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  }
  catch(CDBException *e)
  {
    temp_strm ≪ "Exception␣when␣calling␣‘content_summaries.Open’." ≪ endl ≪
        "Error␣code␣=␣" ≪ e→m_nRetCode ≪ endl ≪ "Exception␣==␣" ≪ e→m_strError ≪ endl ≪
        "SQL␣Code:" ≪ endl ≪ sql_strm.str( );
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
    e→Delete( );
  }    /* catch */
  sql_strm.str("");
```

897. No entries found. [LDF 2006.09.07.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (content_summaries.IsBOF( )) {
#if 0    /* 1 */
    temp_strm ≪ "No␣entries␣in␣‘Content_Summaries’␣with␣" ≪ "‘record_id’␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Content_Summaries_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Content_Summaries␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }    /* if (content_summaries.IsBOF( )) */
```

898. Found entries. [LDF 2006.09.07.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* Found entries in Content_Summaries. */
  { entry_ctr = 0;
```

899.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  long continuation_ctr;
  long content_summary_id_start;
  long content_summary_id_end;
  stringstream content_summary_strm;
  html_strm ≪ "<table␣border=\"1\">\n<caption␣align=\"left\">" ≪
      "<a␣name=\"Content_Summaries_Table_" ≪ records.m_record_id ≪
      "\">␣" ≪ "<b>Content␣Summaries␣Table␣" ≪ records.m_record_id ≪
      "</b></a>\n</caption>\n" ≪ "<tr><th>&nbsp;<b>content_summary_id</b>&nbsp;" ≪
      "<th>&nbsp;<b>record_id</b>&nbsp;" ≪ "<th>&nbsp;<b>continuation</b>&nbsp;" ≪
      "<th>&nbsp;<b>content_summary</b>&nbsp;";
```

900.

⟨ Define **DB_Display** functions 785 ⟩ +≡
```
  while (¬content_summaries.IsEOF()) {
```

901.

⟨ Define **DB_Display** functions 785 ⟩ +≡
```
  if (content_summaries.m_continuation ≡ 0) {
    html_strm ≪ "<tr><td>&nbsp;" ≪ content_summaries.m_content_summary_id ≪
        "&nbsp;\n" ≪ "<td>&nbsp;" ≪ content_summaries.m_record_id ≪ "&nbsp;\n" ≪
        "<td>&nbsp;" ≪ content_summaries.m_continuation ≪ "&nbsp;\n" ≪ "<td>&nbsp;" ≪
        content_summaries.m_content_summary ≪ "&nbsp;\n";
    content_summaries.MoveNext();
  }    /* if (content_summaries.m_continuation ≡ 0) */
```

902.

⟨ Define **DB_Display** functions 785 ⟩ +≡
```
  else if (content_summaries.m_continuation ≡ 1) { content_summary_id_start =
      content_summaries.m_content_summary_id;
  content_summary_strm ≪ content_summaries.m_content_summary;
```

903.

⟨ Define **DB_Display** functions 785 ⟩ +≡
```
  for ( ; ; ) { content_summaries.MoveNext();
```

904.

⟨ Define **DB_Display** functions 785 ⟩ +≡
```
  if (content_summaries.IsEOF() ∨ content_summaries.m_continuation ≡
        0 ∨ content_summaries.m_continuation ≡ 1) {
    html_strm ≪ "<tr><td>&nbsp;" ≪ content_summary_id_start ≪ "&ndash;" ≪
        content_summary_id_end ≪ "&nbsp;\n" ≪ "<td>&nbsp;" ≪ content_summaries.m_record_id ≪
        "&nbsp;\n" ≪ "<td>&nbsp;1&ndash;" ≪ continuation_ctr ≪ "&nbsp;\n" ≪ "<td>&nbsp;" ≪
        content_summary_strm.str() ≪ "&nbsp;\n";
    content_summary_id_start = 0;
    content_summary_id_end = 0;
    content_summary_strm.str("");
    break;
  }
```

905.

⟨ Define **DB_Display** functions 785 ⟩ +≡
```
  else {
    continuation_ctr = content_summaries.m_continuation;
    content_summary_id_end = content_summaries.m_content_summary_id;
    content_summary_strm ≪ content_summaries.m_content_summary;
    continue;
  }
```

906.

⟨ Define **DB_Display** functions 785 ⟩ +≡
```
  }    /* for */
```

907.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  }     /* else if (content_summaries.m_continuation ≡ 1) */
```

908.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  }     /* while */
```

909.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  html_strm ≪ "</table>\n<br>\n"; }     /* else (¬content_summaries.IsBOF()) */
```

910.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  content_summaries.Close();
```

911. **Records_Languages** Vertical table, separate database table for *record_id*. [LDF 2006.08.25.]

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  sql_strm ≪ "select␣*␣from␣Records_Languages␣where␣record_id␣=␣" ≪ record_number;
  records_languages.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

912. No entries found. [LDF 2006.08.25.]

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  if (records_languages.IsBOF()) {
#if 0     /* 1 */
    temp_strm ≪ "No␣entries␣in␣‘Records_Languages’␣with␣" ≪ "‘record_id’␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Languages_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Languages␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }     /* if (records_languages.IsBOF()) */
```

913. Found entries. [LDF 2006.08.25.]

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  else     /* Found entries in Records_Languages. */
  { entry_ctr = 0;
  id_vector.clear();
```

914.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
  while (¬records_languages.IsEOF()) {
    id_vector.push_back(records_languages.m_language_id);
    records_languages.MoveNext();
    ++entry_ctr;
  }     /* while */
```

915.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0    /* 1 */
  temp_strm ≪ "‘id_vector’:" ≪ endl;
#endif
  id_vector_iter = id_vector.begin();
  sql_strm ≪ "select␣*␣from␣Languages␣where␣language_id␣=␣" ≪ *id_vector_iter++;
  for ( ; id_vector_iter ≠ id_vector.end(); ++id_vector_iter) {
#if 0    /* 1 */
    temp_strm ≪ *id_vector_iter ≪ endl;
#endif
    sql_strm ≪ "␣or␣language_id␣=␣" ≪ *id_vector_iter;
  }    /* for */
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  languages.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

916.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (languages.IsBOF()) {
    temp_strm ≪ "WARNING!␣␣In␣‘DB_Display::display_single_record’:" ≪ endl ≪
        "No␣corresponding␣entries␣in␣the␣‘Languages’␣table." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
  }    /* if (languages.IsBOF()) */
```

917.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* ¬languages.IsBOF() */
  {
```

918.

Log

[LDF 2006.08.30.] !! BUG FIX: Removed caption from table. It was redundant. Replaced it with an anchor in a paragraph preceding the table or tables.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<p>\n<a␣name=\"Languages_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
      "<b>Languages_Table␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  while (¬languages.IsEOF()) {
    html_strm ≪ "<table␣border=\"1\">\n" ≪ "<tr><td>&nbsp;<b>language_id</b>&nbsp;" ≪
        "<td>&nbsp;" ≪ languages.m_language_id ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>association_type</b>&nbsp;" ≪ "<td>&nbsp;" ≪
        records_languages.m_association_type ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>association_type_name</b>&nbsp;" ≪
        "<td>&nbsp;" ≪ records_languages.m_association_type_name ≪
        "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>language_name_english</b>&nbsp;" ≪
        "<td>&nbsp;" ≪ languages.m_language_name_english ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>language_name_german</b>&nbsp;" ≪
        "<td>&nbsp;" ≪ languages.m_language_name_german ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>language_abbrev</b>&nbsp;" ≪ "<td>&nbsp;" ≪
        languages.m_language_abbrev ≪ "&nbsp;\n";
    html_strm ≪ "</table>\n<br>\n";
    languages.MoveNext();
  }   /* while */
```

919.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }   /* else (¬languages.IsBOF()) */
```

920.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  languages.Close();
```

921.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }   /* else. Found entries in Languages_Records. */
  records_languages.Close();
```

922. Records_Subjects [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣Records_Subjects␣where␣record_id␣=␣" ≪ record_number;
  records_subjects.Open(CRecordset :: dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

923. No entries found. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (records_subjects.IsBOF()) {
#if 0    /* 1 */
    temp_strm ≪ "No␣entries␣in␣‘Records_Subjects’␣with␣" ≪ "‘record_id’␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Subjects_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Subjects␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }    /* if (records_subjects.IsBOF()) */
```

924. Found entries. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* Found entries in Records_Subjects. */
  { entry_ctr = 0;
  id_vector.clear();
```

925.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬records_subjects.IsEOF()) {
    id_vector.push_back(records_subjects.m_subject_id);
    records_subjects.MoveNext();
    ++entry_ctr;
  }    /* while */
```

926.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0    /* 1 */
  temp_strm ≪ "‘id_vector’:" ≪ endl;
#endif
  id_vector_iter = id_vector.begin();
  sql_strm ≪ "select␣*␣from␣Subjects␣where␣subject_id␣=␣" ≪ *id_vector_iter++;
  for ( ; id_vector_iter ≠ id_vector.end(); ++id_vector_iter) {
#if 0    /* 1 */
    temp_strm ≪ *id_vector_iter ≪ endl;
#endif
    sql_strm ≪ "␣or␣subject_id␣=␣" ≪ *id_vector_iter;
  }    /* for */
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  subjects.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

927.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (subjects.IsBOF( )) {
    temp_strm ≪ "WARNING!␣␣In␣'DB_Display::display_single_record':" ≪ endl ≪
        "No␣corresponding␣entries␣in␣the␣'Subjects'␣table." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
  }    /* if (subjects.IsBOF( )) */
```

928.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else     /* ¬subjects.IsBOF( ) */
  {
```

929.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<p>\n<a␣name=\"Subjects_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
      "<b>Subjects␣Table␣" ≪ records.m_record_id ≪ "</b></a>\n</p>\n\n";
  while (¬subjects.IsEOF( )) {
    html_strm ≪ "<table␣border=\"1\">\n" ≪ "<tr><td>&nbsp;<b>subject_id</b>&nbsp;" ≪
        "<td>&nbsp;" ≪ subjects.m_subject_id ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>subject_type_id</b>&nbsp;" ≪ "<td>&nbsp;" ≪
        subjects.m_subject_type_id ≪ "&nbsp;\n";
    sql_strm ≪ "select␣*␣from␣Subject_Types␣where␣subject_type_id␣=␣" ≪
        subjects.m_subject_type_id;
    subject_types.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
    sql_strm.str("");
    html_strm ≪ "<tr><td>&nbsp;<b>subject_type␣English␣description</b>␣" ≪
        "(from␣<b>Subject_Types</b>␣table)" ≪ "&nbsp;" ≪ "<td>&nbsp;";
    if (subject_types.IsBOF( )) html_strm ≪ "N/A";
    else html_strm ≪ subject_types.m_description_english;
    html_strm ≪ "&nbsp;\n";
    html_strm ≪ "<tr><td>&nbsp;<b>subject_type␣German␣description</b>␣" ≪
        "(from␣<b>Subject_Types</b>␣table)" ≪ "&nbsp;" ≪ "<td>&nbsp;";
    if (subject_types.IsBOF( )) html_strm ≪ "N/A";
    else html_strm ≪ subject_types.m_description_german;
    html_strm ≪ "&nbsp;\n";
    subject_types.Close( );
    html_strm ≪ "<tr><td>&nbsp;<b>subject</b>&nbsp;" ≪ "<td>&nbsp;" ≪ subjects.m_subject ≪
        "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>id_number_ppn</b>&nbsp;" ≪ "<td>&nbsp;" ≪
        subjects.m_id_number_ppn ≪ "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>chain_number</b>&nbsp;" ≪
        "<td>&nbsp;" ≪ subjects.m_chain_number ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>chain_link_number</b>&nbsp;" ≪ "<td>&nbsp;" ≪
        subjects.m_chain_link_number ≪ "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>chain_info</b>&nbsp;" ≪
        "<td>&nbsp;" ≪ subjects.m_chain_info ≪ "&nbsp;\n" ≪ "</table>\n<br>\n\n";
    subjects.MoveNext( );
  }    /* while */
```

930.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else (¬subjects.IsBOF( )) */
```

931.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  subjects.Close( );
```

932.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else Found entries in Subjects_Records. */
  records_subjects.Close( );
```

933. Permutation_Patterns [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣Permutation_Patterns␣where␣record_id␣=␣" ≪ record_number;
  permutation_patterns.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

934. No entries found. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (permutation_patterns.IsBOF( )) {
#if 0    /* 1 */
    temp_strm ≪ "No␣entries␣in␣‘Permutation_Patterns’␣with␣" ≪ "‘record_id’␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Permutation_Patterns_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Permutation_Patterns␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }    /* if (permutation_patterns.IsBOF( )) */
```

935. Found entries. [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* Found entries in Permutation_Patterns. */
  { entry_ctr = 0;
```

936.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<table␣border=\"1\">\n<caption␣align=\"left\">" ≪
      "<a␣name=\"Permutation_Patterns_Table_" ≪ records.m_record_id ≪
      "\">␣" ≪ "<b>Permutation␣Patterns␣Table␣" ≪ records.m_record_id ≪
      "</b></a>\n</caption>\n" ≪ "<tr><th>&nbsp;<b>permutation_pattern_id</b>&nbsp;" ≪
      "<th>&nbsp;<b>record_id</b>&nbsp;" ≪ "<th>&nbsp;<b>subject_id_start</b>&nbsp;" ≪
      "<th>&nbsp;<b>subject_id_end</b>&nbsp;" ≪ "<th>&nbsp;<b>chain_number</b>&nbsp;" ≪
      "<th>&nbsp;<b>permutation_pattern</b>&nbsp;";
  while (¬permutation_patterns.IsEOF( )) {
    html_strm ≪ "<tr><td>&nbsp;" ≪ permutation_patterns.m_permutation_pattern_id ≪
        "&nbsp;\n" ≪ "<td>&nbsp;" ≪ permutation_patterns.m_record_id ≪ "&nbsp;\n" ≪
        "<td>&nbsp;" ≪ permutation_patterns.m_subject_id_start ≪ "&nbsp;\n" ≪
        "<td>&nbsp;" ≪ permutation_patterns.m_subject_id_end ≪ "&nbsp;\n" ≪
        "<td>&nbsp;" ≪ permutation_patterns.m_chain_number ≪ "&nbsp;\n" ≪ "<td>&nbsp;" ≪
        permutation_patterns.m_permutation_pattern ≪ "&nbsp;\n";
    permutation_patterns.MoveNext( );
  }    /* while */
```

937.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "</table>\n<br>\n"; }      /* else (¬permutation_patterns.IsBOF()) */
```

938.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  permutation_patterns.Close();
```

939. Records_Physical_Descriptions Vertical table, separate database table for *record_id*. [LDF 2006.08.25.] ▮

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣Records_Physical_Descriptions␣" ≪ "where␣record_id␣=␣" ≪
      record_number;
  records_physical_descriptions.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

940. No entries found. [LDF 2006.08.25.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (records_physical_descriptions.IsBOF()) {
#if 0     /* 1 */
    temp_strm ≪ "No␣entries␣in␣'Records_Physical_Descriptions'␣with␣" ≪ "'record_id'␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Physical_Descriptions_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Physical_Descriptions␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }     /* if (records_physical_descriptions.IsBOF()) */
```

941. Found entries. [LDF 2006.08.25.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else     /* Found entries in Records_Physical_Descriptions. */
  { entry_ctr = 0;
  id_vector.clear();
```

942.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬records_physical_descriptions.IsEOF()) {
    id_vector.push_back(records_physical_descriptions.m_physical_description_id);
    records_physical_descriptions.MoveNext();
    ++entry_ctr;
  }     /* while */
```

943.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0    /* 1 */
  temp_strm ≪ "‘id_vector’:" ≪ endl;
#endif
  id_vector_iter = id_vector.begin( );
  sql_strm ≪ "select␣*␣from␣Physical_Descriptions␣where␣physical_description_id␣=␣" ≪
      *id_vector_iter++;
  for ( ; id_vector_iter ≠ id_vector.end( ); ++id_vector_iter) {
#if 0    /* 1 */
    temp_strm ≪ *id_vector_iter ≪ endl;
#endif
    sql_strm ≪ "␣or␣physical_description_id␣=␣" ≪ *id_vector_iter;
  }    /* for */
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str( );
  AfxMessageBox(temp_strm.str( ).c_str( ));
  temp_strm.str("");
#endif
  physical_descriptions.Open(CRecordset :: dynaset, sql_strm.str( ).c_str( ));
  sql_strm.str("");
```

944.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (physical_descriptions.IsBOF( )) {
    temp_strm ≪ "WARNING!␣␣In␣‘DB_Display::display_single_record’:" ≪ endl ≪
        "No␣corresponding␣entries␣in␣the␣‘Physical_Descriptions’␣table." ≪ endl ≪
        "Continuing.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
  }    /* if (physical_descriptions.IsBOF( )) */
```

945.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* ¬physical_descriptions.IsBOF( ) */
  {
```

946.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬physical_descriptions.IsEOF( )) { html_strm ≪ "<table␣border=\"1\">\n" ≪
      "<caption␣align=\"left\">" ≪ "<a␣name=\"Physical_Descriptions_Table_" ≪
      records.m_record_id ≪ "\">␣" ≪ "<b>Physical␣Descriptions␣Table␣" ≪ records.m_record_id ≪
      "&nbsp;</b></a></caption>\n" ≪ "<tr><td>&nbsp;<b>physical_description_id</b>&nbsp;" ≪
      "<td>&nbsp;" ≪ physical_descriptions.m_physical_description_id ≪ "&nbsp;\n" ≪
      "<tr><td>&nbsp;<b>text</b>&nbsp;" ≪ "<td>&nbsp;" ≪ physical_descriptions.m_text ≪
      "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>pica_category_id</b>&nbsp;" ≪ "<td>&nbsp;" ≪
      physical_descriptions.m_pica_category_id ≪ "&nbsp;\n";
```

947. PICA Categories. [LDF 2006.08.25.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣PICA_Categories␣" ≪ "where␣pica_category_id␣=␣" ≪
      physical_descriptions.m_pica_category_id;
  pica_categories.Open(CRecordset :: dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

948.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (pica_categories.IsBOF()) {
    html_strm ≪ "<tr><td␣␣colspan=\"2\">" ≪ "<font␣color=\"red\">" ≪
        "&nbsp;<b>Invalid␣PICA␣Category</b>&nbsp;</font>\n";
  }
```

949.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else {
    html_strm ≪ "<tr><td>&nbsp;<b>Pica+␣Category␣Code</b>&nbsp;<br>" ≪
        "&nbsp;(From␣<b>PICA_Categories</b>␣table)&nbsp;" ≪
        "<td>&nbsp;" ≪ pica_categories.m_pica_plus_category_code ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>Pica3␣Category␣Code</b>&nbsp;<br>" ≪
        "&nbsp;(From␣<b>PICA_Categories</b>␣table)&nbsp;" ≪
        "<td>&nbsp;" ≪ pica_categories.m_pica_3_category_code ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>PICA␣Category,␣English␣Description</b>" ≪
        "&nbsp;<br>" ≪ "&nbsp;(From␣<b>PICA_Categories</b>␣table)&nbsp;" ≪
        "<td>&nbsp;" ≪ pica_categories.m_description_english ≪ "&nbsp;\n" ≪
        "<tr><td>&nbsp;<b>PICA␣Category,␣German␣Description</b>" ≪ "&nbsp;<br>" ≪
        "&nbsp;(From␣<b>PICA_Categories</b>␣table)&nbsp;" ≪ "<td>&nbsp;" ≪
        pica_categories.m_description_german ≪ "&nbsp;\n";
  }
```

950. End of PICA Categories. [LDF 2006.08.25.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  pica_categories.Close();
  html_strm ≪ "<tr><td>&nbsp;<b>pica_field_id</b>&nbsp;" ≪ "<td>&nbsp;" ≪
      physical_descriptions.m_pica_field_id ≪ "&nbsp;\n";
```

951. PICA Fields. [LDF 2006.08.25.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣PICA_Fields␣where␣pica_field_id␣=␣" ≪
      physical_descriptions.m_pica_field_id;
  pica_fields.Open(CRecordset :: dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

952.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (pica_fields.IsBOF()) {
    html_strm ≪ "<tr><td␣␣colspan=\"2\">" ≪ "<font␣color=\"red\">" ≪
        "&nbsp;<b>Invalid␣PICA␣Field</b>&nbsp;</font>\n";
  }
```

953.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
else {
  html_strm ≪ "<tr><td>&nbsp;<b>Pica+␣Field␣Code</b>&nbsp;<br>" ≪
      "&nbsp;(From␣<b>PICA_Fields</b>␣table)&nbsp;" ≪ "<td>&nbsp;" ≪
      pica_fields.m_pica_plus_field_code ≪ "&nbsp;\n" ≪ "<tr><td>&nbsp;<b>Pi\
      ca3␣Field␣Code</b>&nbsp;<br>" ≪ "&nbsp;(From␣<b>PICA_Fields</b>␣table)&nbsp;" ≪
      "<td>&nbsp;" ≪ pica_fields.m_pica_3_field_code ≪ "&nbsp;\n" ≪
      "<tr><td>&nbsp;<b>PICA␣Field,␣English␣Description</b>&nbsp;<br>" ≪
      "&nbsp;(From␣<b>PICA_Fields</b>␣table)&nbsp;" ≪
      "<td>&nbsp;" ≪ pica_fields.m_description_english ≪ "&nbsp;\n" ≪
      "<tr><td>&nbsp;<b>PICA␣Field,␣German␣Description</b>&nbsp;<br>" ≪
      "&nbsp;(From␣<b>PICA_Fields</b>␣table)&nbsp;" ≪ "<td>&nbsp;" ≪
      pica_fields.m_description_german ≪ "&nbsp;\n";
}
```

954. End of PICA Fields. [LDF 2006.08.25.]

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
pica_fields.Close();
```

955.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
html_strm ≪ "</table>\n<br>\n";
physical_descriptions.MoveNext(); }      /* while */
```

956.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
}      /* else (¬physical_descriptions.IsBOF()) */
```

957.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
physical_descriptions.Close();
```

958.

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
}      /* else Found entries in Physical_Descriptions_Records. */
records_physical_descriptions.Close();
```

959. **Records_Publishers** Horizontal table, separate database table for *record_id*. [LDF 2006.08.28.]

⟨ Define **DB_Display** functions 785 ⟩ +≡

```
sql_strm ≪ "select␣*␣from␣Records_Publishers␣where␣record_id␣=␣" ≪ record_number;
records_publishers.Open(CRecordset::dynaset, sql_strm.str().c_str());
sql_strm.str("");
```

960. No entries found. [LDF 2006.08.28.] k

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (records_publishers.IsBOF()) {
#if 0     /* 1 */
    temp_strm ≪ "No␣entries␣in␣‘Records_Publishers’␣with␣" ≪ "‘record_id’␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Publishers_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Publishers␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }     /* if (records_publishers.IsBOF()) */
```

961. Found entries. [LDF 2006.08.28.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else     /* Found entries in Records_Publishers. */
  { html_strm ≪ "<table␣border=\"1\"><caption␣align=\"left\">" ≪
      "<a␣name=\"Publishers_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
      "<b>Publishers␣Table␣" ≪ records.m_record_id ≪ "&nbsp;</b></a></caption>\n" ≪
      "<tr><th>&nbsp;<b>publisher_id</b>&nbsp;<th>&nbsp;" ≪
      "<b>publisher_name</b>&nbsp;<th>&nbsp;<b>place</b>&nbsp;" ≪
      "<th>&nbsp;<b>primary_info_source</b>&nbsp;";
  entry_ctr = 0;
  id_vector.clear();
```

962.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬records_publishers.IsEOF()) {
    id_vector.push_back(records_publishers.m_publisher_id);
    records_publishers.MoveNext();
    ++entry_ctr;
  }     /* while */
```

963.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0    /* 1 */
  temp_strm ≪ "'id_vector':" ≪ endl;
#endif
  id_vector_iter = id_vector.begin();
  sql_strm ≪ "select␣*␣from␣Publishers␣where␣publisher_id␣=␣" ≪ *id_vector_iter++;
  for ( ; id_vector_iter ≠ id_vector.end(); ++id_vector_iter) {
#if 0    /* 1 */
    temp_strm ≪ *id_vector_iter ≪ endl;
#endif
    sql_strm ≪ "␣or␣publisher_id␣=␣" ≪ *id_vector_iter;
  }    /* for */
#if 0    /* 1 */
  temp_strm ≪ "SQL␣Code:\n" ≪ sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  publishers.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

964.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (publishers.IsBOF()) {
    temp_strm ≪ "WARNING!␣␣In␣'DB_Display::display_single_record':" ≪ endl ≪
        "No␣corresponding␣entries␣in␣the␣'Publishers'␣table." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
  }    /* if (publishers.IsBOF()) */
  else    /* ¬publishers.IsBOF() */
  {
```

965.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬publishers.IsEOF()) {
    html_strm ≪ "\n<tr><td>" ≪ publishers.m_publisher_id ≪ "<td>" ≪ publishers.m_publisher_name ≪
        "&nbsp;" ≪ "<td>" ≪ publishers.m_place ≪ "&nbsp;" ≪ "<td>" ≪
        publishers.m_primary_info_source ≪ "&nbsp;";
    publishers.MoveNext();
  }    /* while */
```

966.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else (¬publishers.IsBOF()) */
```

967.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<br>\n</table>\n<br>\n";
  publishers.Close();
```

968.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else Found entries in Records_Publishers. */
  records_publishers.Close();
```

969. Records_Database_Providers Horizontal table, separate database table for *record_id*. [LDF 2006.08.28.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  sql_strm ≪ "select␣*␣from␣Records_Database_Providers␣" ≪ "where␣record_id␣=␣" ≪
      record_number;
  records_database_providers.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

970. No entries found. [LDF 2006.08.28.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (records_database_providers.IsBOF()) {
#if 0    /* 1 */
    temp_strm ≪ "No␣entries␣in␣‘Records_Database_Providers’␣with␣" ≪ "‘record_id’␣==␣" ≪
        record_number ≪ "." ≪ endl ≪ "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    html_strm ≪ "<p>\n<a␣name=\"Database_Providers_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
        "<b>No␣Database␣Providers␣for␣Record␣" ≪ records.m_record_id ≪ "</b></a></p>\n\n";
  }    /* if (records_database_providers.IsBOF()) */
```

971. Found entries. [LDF 2006.08.28.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* Found entries in Records_Database_Providers. */
  { html_strm ≪ "<table␣border=\"1\"><caption␣align=\"left\">" ≪
      "<a␣name=\"Database_Providers_Table_" ≪ records.m_record_id ≪ "\">␣" ≪
      "<b>Database␣Providers␣Table␣" ≪ records.m_record_id ≪ "&nbsp;</b></a></caption>\n" ≪
      "<tr><th>&nbsp;<b>database_provider_id</b>&nbsp;<th>&nbsp;" ≪
      "<b>database_provider_name</b>&nbsp;<th>&nbsp;<b>place</b>&nbsp;";
  entry_ctr = 0;
  id_vector.clear();
```

972.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬records_database_providers.IsEOF()) {
    id_vector.push_back(records_database_providers.m_database_provider_id);
    records_database_providers.MoveNext();
    ++entry_ctr;
  }    /* while */
```

973.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0    /* 1 */
  temp_strm << "‘id_vector’:" << endl;
#endif
  id_vector_iter = id_vector.begin();
  sql_strm << "select␣*␣from␣Database_Providers␣where␣database_provider_id␣=␣" <<
      *id_vector_iter++;
  for ( ; id_vector_iter ≠ id_vector.end(); ++id_vector_iter) {
#if 0    /* 1 */
    temp_strm << *id_vector_iter << endl;
#endif
    sql_strm << "␣or␣database_provider_id␣=␣" << *id_vector_iter;
  }    /* for */
#if 0    /* 1 */
  temp_strm << "SQL␣Code:\n" << sql_strm.str();
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  database_providers.Open(CRecordset::dynaset, sql_strm.str().c_str());
  sql_strm.str("");
```

974.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (database_providers.IsBOF()) {
    temp_strm << "WARNING!␣␣In␣‘DB_Display::display_single_record’:" << endl <<
        "No␣corresponding␣entries␣in␣the␣‘Database_Providers’␣table." << endl <<
        "Continuing.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
  }    /* if (database_providers.IsBOF()) */
```

975.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  else    /* ¬database_providers.IsBOF() */
  {
```

976.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  while (¬database_providers.IsEOF()) {
    html_strm << "\n<tr><td>" << database_providers.m_database_provider_id << "<td>" <<
        database_providers.m_database_provider_name << "&nbsp;" << "<td>" <<
        database_providers.m_place << "&nbsp;";
    database_providers.MoveNext();
  }    /* while */
```

977.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  }    /* else (¬database_providers.IsBOF()) */
```

978.

⟨ Define **DB_Display** functions 785 ⟩ +≡
```
  html_strm ≪ "<br>\n</table>\n<br>\n";
  database_providers.Close( );
```

979.

⟨ Define **DB_Display** functions 785 ⟩ +≡
```
  }    /* else Found entries in Records_Database_Providers. */
  records_database_providers.Close( );
```

980.

⟨ Define **DB_Display** functions 785 ⟩ +≡
```
  return 0; }    /* End of DB_Display :: display_single_record definition. */
```

981. Display records. [LDF 2006.08.23.]
display_records displays a range of records from record number *start* to record number *end*. Both 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 $0 < end < start$, 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. [LDF 2006.08.23.]

Log

[LDF 2006.08.23.] Added this function.

[[LDF 2006.09.06.]] Added the optional **CString** *search_command* argument with the default "".

⟨ Declare **DB_Display** functions 784 ⟩ +≡
```
  int display_records(CString search_command_str = "", unsigned int start = 1, unsigned int end = 0);
```

982.

⟨ Define **DB_Display** functions 785 ⟩ +≡
```
  int DB_Display :: display_records(CString search_command_str, unsigned int start, unsigned int
        end){ stringstream temp_strm;
#if 0    /* 1 */
    temp_strm ≪ "In␣‘DB_Display::display_records’." ≪ endl ≪ "‘search_command_str’␣==␣" ≪
        search_command_str;
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
```

983. Error handling: *html_strm* is closed. Can't display records. [LDF 2006.08.23.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (¬html_strm.is_open()) {
    temp_strm ≪ "ERROR!␣␣In␣‘DB_Display::display_records’:" ≪ endl ≪
        "‘DB_Display::html_strm’␣isn’t␣open.␣␣Can’t␣display␣records." ≪ endl ≪
        "Exiting␣function␣unsuccessfully␣with␣return␣value␣1.";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    return 1;
  }    /* if (¬html_strm.is_open()) */
```

984.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  int r = 0;
```

985. Warning: $end > 0 \wedge end < start$. [LDF 2006.08.23.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  if (end > 0 ∧ end < start) {
    temp_strm ≪ "WARNING!␣␣In␣‘DB_Display::display_records’:" ≪
        endl ≪ "‘end␣>␣0␣&&␣end␣<␣start’" ≪ endl ≪
        "This␣isn’t␣allowed.␣␣Will␣display␣all␣records␣from␣" ≪ "‘start’␣==␣" ≪ start ≪ ".";
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
    end = 0;
  }    /* if (end > 0 ∧ end < start) */
```

986.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  int record_ctr;
  vector⟨long⟩ record_id_vector;
  record_ctr = count_records(start, end, &record_id_vector);
```

987.

Log

[LDF 2006.09.06.] Added code for displaying *search_command_str*, if present.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0    /* 1 */
  temp_strm ≪ "‘record_ctr’␣==␣" ≪ record_ctr ≪ endl ≪ "Record␣IDs:" ≪ endl;
  for (vector⟨long⟩::iterator iter = record_id_vector.begin(); iter ≠ record_id_vector.end(); ++iter)
    temp_strm ≪ *iter ≪ endl;
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  html_strm ≪ "<p>\n" ≪ record_ctr ≪ "␣record";
  if (record_ctr ≠ 1) html_strm ≪ "s";
  html_strm ≪ "␣found.\n<br>\n<br>\n";
  if (search_command_str ≡ "") {
    html_strm ≪ "No␣search␣command.\n<br>\n";
  }
  else {
    html_strm ≪ search_command_str;
  }
  html_strm ≪ "</p>\n\n" ≪ "<hr␣size=\"2\"␣color=\"black\">\n\n";
```

988. Table of Contents.

Log

[LDF 2006.09.06.] Added this section.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<h2␣align=\"center\"><a␣name=\"Table_of_Contents\">␣" ≪
      "Table␣of␣Contents␣</a>\n</h2>\n\n" ≪ "<table>\n" ≪
      "<tr><td><a␣href=\"#Record_List_No_Tables\">␣" ≪ "Record␣List</a>\n" ≪
      "<tr><td><a␣href=\"#Record_List_With_Tables\">␣" ≪ "Record␣List␣With␣Tables</a>\n" ≪
      "<tr><td><a␣href=\"#Records_Complete\">␣" ≪ "Records␣Complete</a>\n" ≪
      "</table>\n\n" ≪ "<hr␣size=\"2\"␣color=\"black\">\n\n" ≪
      "<p>\n<a␣href=\"#Top\">␣Back␣to␣Top</a>\n</p>\n\n" ≪
      "<hr␣size=\"2\"␣color=\"black\">\n\n";
```

989. Write a list of links to the individual records, without listing their tables. [LDF 2006.09.06.]

Log

[LDF 2006.09.06.] Added this section.

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<h2␣align=\"center\"><a␣name=\"" ≪ "Record_List_No_Tables\">␣" ≪
      "Record␣List␣</a>\n</h2>\n\n" ≪ "<table>\n";
  int i = 1;
  for (vector⟨long⟩::iterator iter = record_id_vector.begin(); iter ≠ record_id_vector.end(); ++iter) {
    if (i > 1) html_strm ≪ "<br>\n";
    html_strm ≪ "<tr><td><a␣href=\"#Record_" ≪ *iter ≪ "\">␣" ≪ i ≪ ".␣␣Record␣" ≪ *iter ≪
        "</a>\n";
    ++i;
  }   /* for */
  html_strm ≪ "</table><br>\n<hr␣size=\"2\"␣color=\"black\">\n\n" ≪
      "<p>\n<a␣href=\"#Table_of_Contents\">␣" ≪ "Back␣to␣Table␣of␣Contents</a>\n<br>\n" ≪
      "<a␣href=\"#Top\">␣Back␣to␣Top</a>\n</p>\n\n" ≪
      "<hr␣size=\"2\"␣color=\"black\">\n\n";
```

990. Write a list of links to the individual records, and their tables. They are of the form "⟨*number*⟩ Record ⟨*record_id*⟩", e.g.,

1. Record 37
   Records Table
   Access Numbers Table
   etc.
2. Record 38
   Records Table
   Access Numbers Table
   etc.
3. Record 39
   Records Table
   Access Numbers Table
   etc.

(etc.)

[LDF 2006.08.23.] [LDF 2006.08.24.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
 html_strm ≪ "<h2␣align=\"center\"><a␣name=\"Record_List_With_Tables\">␣" ≪
     "Record␣List␣With␣Tables</a>\n</h2>\n\n" ≪ "<table>\n";
 for (int j = 1; j < i; ++j) {
   if (j > 1) html_strm ≪ "<br>\n";
   html_strm ≪ "<tr><td><a␣href=\"#Record_" ≪ j ≪ "\">␣" ≪ j ≪ ".␣␣Record␣" ≪ j ≪ "</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Records_Table_" ≪ j ≪ "\">␣" ≪ "Records␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Call_Numbers_Table_" ≪ j ≪ "\">␣" ≪
       "Call␣Numbers␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Access_Numbers_Table_" ≪ j ≪ "\">␣" ≪
       "Access␣Numbers␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Remote_Access_Table_" ≪ j ≪ "\">␣" ≪
       "Remote␣Access␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Exemplar_Production_Numbers_Table_" ≪ "\">␣" ≪
       "Exemplar␣Production␣Numbers␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Bibliographic_Types_Table_" ≪ j ≪ "\">␣" ≪
       "Bibliographic␣Types␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Authors_Table_" ≪ j ≪ "\">␣" ≪ "Authors␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Contributors_Table_" ≪ j ≪ "\">␣" ≪
       "Contributors␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Main_Titles_Table_" ≪ j ≪ "\">␣" ≪ "Main␣Titles␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Content_Summaries_Table_" ≪ j ≪ "\">␣" ≪
       "Content␣Summaries␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Languages_Table_" ≪ j ≪ "\">␣" ≪ "Languages␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Subjects_Table_" ≪ j ≪ "\">␣" ≪ "Subjects␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Permutation_Patterns_Table_" ≪ j ≪ "\">␣" ≪
       "Permutation␣Patterns␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Physical_Descriptions_Table_" ≪ j ≪ "\">␣" ≪
       "Physical␣Descriptions␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Publishers_Table_" ≪ j ≪ "\">␣" ≪ "Publishers␣Table</a>\n"
    ≪ "<tr><td><td><a␣href=\"#Database_Providers_Table_" ≪ j ≪ "\">␣" ≪
       "Database␣Providers␣Table</a>\n";
 }  /* for */
```

```
  html_strm ≪ "</table><br>\n<hr␣size=\"2\"␣color=\"black\">\n\n" ≪ "<p>\n" ≪
      "<a␣href=\"#Record_List_No_Tables\">␣Back␣to␣Record␣List</a>\n<br>\n" ≪
      "<a␣href=\"#Table_of_Contents\">␣Back␣to␣Table␣of␣Contents</a>\n<br>\n" ≪
      "<a␣href=\"#Top\">␣Back␣to␣Top</a>\n</p>\n\n" ≪
      "<hr␣size=\"2\"␣color=\"black\">\n\n";
```

991. Display records until *display_single_record* returns a non-zero value. PLEASE NOTE: This will fail to display all the records, if there's a gap in the sequence of *record_ids*. [LDF 2006.08.23.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  html_strm ≪ "<h2␣align=\"center\"><a␣name=\"Records_Complete\">␣" ≪
      "Records</a>\n</h2>\n\n";
  record_ctr = 0;
  if (end ≡ 0) {
#if 0     /* 1 */
    temp_strm ≪ "Displaying␣all␣records.";
    AfxMessageBox(temp_strm.str( ).c_str( ));
    temp_strm.str("");
#endif
    r = 0;
    while (r ≡ 0) {
      ++record_ctr;
#if 0     /* 1 */
      temp_strm ≪ "‘record_ctr’␣␣==␣" ≪ record_ctr;
      AfxMessageBox(temp_strm.str( ).c_str( ));
      temp_strm.str("");
#endif
      r = display_single_record(start++, record_ctr);
      html_strm ≪ "<p>\n" ≪ "<a␣href=\"#Record_List_With_Tables\">␣" ≪
          "Back␣to␣Record␣List␣with␣Tables</a>\n<br>\n" ≪
          "<a␣href=\"#Record_List_No_Tables\">␣" ≪ "Back␣to␣Record␣List</a>\n<br>\n" ≪
          "<a␣href=\"#Table_of_Contents\">␣" ≪ "Back␣to␣Table␣of␣Contents</a>\n<br>\n" ≪
          "<a␣href=\"#Top\">␣Back␣to␣Top</a>\n</p>\n\n" ≪
          "<hr␣size=\"2\"␣color=\"black\">\n\n";
    }     /* while */
    return 0;
  }     /* if (end ≡ 0) */
```

992. Display records from *start* to *end*. The loop continues until *end* is reached, even if there's no record with *record_id* ≡ *end*.

```
⟨ Define DB_Display functions 785 ⟩ +≡
#if 0    /* 1 */
  temp_strm ≪ "Displaying␣a␣range␣of␣records.";
  AfxMessageBox(temp_strm.str().c_str());
  temp_strm.str("");
#endif
  for ( ; start ≤ end; ++start) {
    ++record_ctr;
#if 0    /* 1 */
    temp_strm ≪ "‘record_ctr’␣␣==␣" ≪ record_ctr;
    AfxMessageBox(temp_strm.str().c_str());
    temp_strm.str("");
#endif
    r = display_single_record(start, record_ctr);
    if (r ≡ 0) html_strm ≪ "<p>\n<a␣href=\"#Top\">␣Back␣to␣Top</a>\n</p>\n\n" ≪
          "<hr␣size=\"2\"␣color=\"black\">\n\n";
  }    /* for */
```

993. End of **DB_Display** :: *display_records* definition. [LDF Undated.]

```
⟨ Define DB_Display functions 785 ⟩ +≡
  return 0; }
```

994. Putting DB_Display together.

995. This is what gets written to `dbdspl.h`.

```
⟨ dbdspl.h 995 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class DB_Display 782 ⟩
```

996. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ dbdspl.web 778 ⟩
⟨ Define DB_Display functions 785 ⟩
```

997. ODBC Classes. [LDF 2006.09.22.]

998. PICA_Categories (picacats.web). [LDF 2006.09.22.]

Log

[LDF 2006.09.22.] Created this file. It contains code formerly in `picacats.h` and `picacats.cpp`.

```
⟨ picacats.web 998 ⟩ ≡
  static char id_string[] = "$Id:␣picacats.web,v␣1.6␣2006/10/23␣14:23:17␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.
This code is used in section 1018.

999. Preprocessor macro calls. [LDF 2006.09.22.]

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1000. using declarations for namespaces. [LDF 2006.09.22.]

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1001. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1002. PICA_Categories class declaration. [LDF 2006.08.25.]
Feld-/Parameterdaten. Die Zeichenfolgentypen reflektieren den eigentlichen Datentyp des Datenbankfelds (**CStringA** für ANSI-Datentypen und **CStringW** für Unicode-Datentypen), um zu verhindern, daß der ODBC-Treiber nicht erforderliche Konvertierungen ausführt. Sie können diese Member zu **CString**-Typen ändern damit der ODBC-Treiber alle erforderlichen Konvertierungen ausführt. Hinweis: Sie müssen mindestens die ODBC-Treiberversion 3.5 verwenden, um Unicode und die Konvertierungen zu unterstützen.

Log

[LDF 2006.08.25.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class PICA_Categories 1002 ⟩ ≡
  class PICA_Categories : public CRecordset {
  public: long m_pica_category_id;
    CStringA m_pica_plus_category_code;
    int m_pica_3_category_code;
    CStringA m_description_german;
    CStringA m_description_english;
    ⟨ Declare PICA_Categories functions 1004 ⟩
  };
```

This code is used in section 1017.

1003. Functions.

1004. Constructor. [LDF 2006.08.25.]

```
⟨ Declare PICA_Categories functions 1004 ⟩ ≡
  PICA_Categories(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(PICA_Categories)
```

See also sections 1006, 1008, 1010, 1012, and 1014.

This code is used in section 1002.

1005.

```
⟨ Define PICA_Categories functions 1005 ⟩ ≡
  PICA_Categories::PICA_Categories(CDatabase *pdb)
  : CRecordset(pdb) {
    m_pica_category_id = 0;
    m_pica_plus_category_code = "";
    m_pica_3_category_code = 0;
    m_description_german = "";
    m_description_english = "";
    m_nFields = 5;
    m_nDefaultType = dynaset;
  }
```

See also sections 1007, 1009, 1011, 1013, and 1015.

This code is used in section 1018.

1006. Get default connect. [LDF 2006.08.25.]

```
⟨ Declare PICA_Categories functions 1004 ⟩ +≡
  virtual CString GetDefaultConnect( );    /* Standard-Verbindungszeichenfolge */
```

1007. Commented-out:
error Sicherheitsproblem: Die Verbindungszeichenfolge enthält möglicherweise ein Kennwort Die Verbindungszeichenfolge enthlt möglicherweise einfache Textkennwörter und/oder andere vertrauliche Informationen. Entfernen Sie # **error**, nachdem Sie die Verbindungszeichenfolge berprüft haben. Sie können das Kennwort in einem anderen Format speichern oder eine andere
Benutzerauthentifizierung verwenden.

Log

[LDF 2006.09.22.] Modified this function. Added **stringstream** *temp_strm*. I had to do this because the literal string passed to _T would have been too long for CWEAVE to cope with.

```
⟨ Define PICA_Categories functions 1005 ⟩ +≡
  CString PICA_Categories::GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1008. Standard-SQL for Recordset. [LDF 2006.08.25.]

```
⟨ Declare PICA_Categories functions 1004 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1009.

```
⟨ Define PICA_Categories functions 1005 ⟩ +≡
  CString PICA_Categories::GetDefaultSQL( )
  {
    return _T("[dbo].[PICA_Categories]");
  }
```

1010. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.25.]

```
⟨ Declare PICA_Categories functions 1004 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1011. Makros, z.B. *RFX_Text*( ) und *RFX_Int*( ), sind vom Typ der Membervariablen abhängig, nicht vom Typ des Felds in der Datenbank. ODBC konvertiert den Spaltenwert automatisch in den angeforderten Typ.

```
⟨ Define PICA_Categories functions 1005 ⟩ +≡
  void PICA_Categories::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[pica_category_id]"), m_pica_category_id);
    RFX_Text(pFX, _T("[pica_plus_category_code]"), m_pica_plus_category_code);
    RFX_Int(pFX, _T("[pica_3_category_code]"), m_pica_3_category_code);
    RFX_Text(pFX, _T("[description_german]"), m_description_german);
    RFX_Text(pFX, _T("[description_english]"), m_description_english);
  }
```

1012. Assert Valid. [LDF 2006.08.25.]

```
⟨ Declare PICA_Categories functions 1004 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1013.

```
⟨ Define PICA_Categories functions 1005 ⟩ +≡
#ifdef _DEBUG
  void PICA_Categories::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif    /* _DEBUG */
```

1014. Dump. [LDF 2006.08.25.]

```
⟨ Declare PICA_Categories functions 1004 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1015.

```
⟨ Define PICA_Categories functions 1005 ⟩ +≡
#ifdef _DEBUG
  void PICA_Categories::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif    /* _DEBUG */
```

1016. Putting PICA_Categories together. [LDF 2006.09.22.]

1017. This is what gets written to `picacats.h`. [LDF 2006.09.22.]

```
⟨ picacats.h 1017 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class PICA_Categories 1002 ⟩
```

1018. This is what gets compiled. [LDF 2006.09.22.]

```
  ⟨ Include files 13 ⟩
  ⟨ picacats.web 998 ⟩
  IMPLEMENT_DYNAMIC(PICA_Categories, CRecordset)
  ⟨ Define PICA_Categories functions 1005 ⟩
```

1019. PICA_Fields (`picaflds.web`).

Log

[LDF 2006.09.25.] Created this file. It contains code formerly in `picaflds.h` and `picaflds.cpp`.

```
⟨ picaflds.web 1019 ⟩ ≡
  static char id_string[] = "$Id:␣picaflds.web,v␣1.6␣2006/10/23␣14:23:25␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1039.

1020. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1021. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1022. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1023. PICA_Fields class declaration.

Log

[LDF 2006.09.25.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class PICA_Fields 1023 ⟩ ≡
  class PICA_Fields : public CRecordset {
  public: long m_pica_field_id;
    CStringA m_pica_plus_field_code;
    CStringA m_pica_3_field_code;
    CStringA m_description_german;
    CStringA m_description_english;
    ⟨ Declare PICA_Fields functions 1025 ⟩
  };
```

This code is used in section 1038.

1024. Functions.

1025. Constructor.

```
⟨ Declare PICA_Fields functions 1025 ⟩ ≡
  PICA_Fields(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(PICA_Fields)
```
See also sections 1027, 1029, 1031, 1033, and 1035.
This code is used in section 1023.

1026.

```
⟨ Define PICA_Fields functions 1026 ⟩ ≡
  IMPLEMENT_DYNAMIC(PICA_Fields, CRecordset)PICA_Fields :: PICA_Fields(CDatabase *pdb)
  : CRecordset(pdb) {
    m_pica_field_id = 0;
    m_pica_plus_field_code = "";
    m_pica_3_field_code = "";
    m_description_german = "";
    m_description_english = "";
    m_nFields = 5;
    m_nDefaultType = dynaset;
  }
```
See also sections 1028, 1030, 1032, 1034, and 1036.
This code is used in section 1039.

1027. Get default connect. [LDF 2006.09.25.]

```
⟨ Declare PICA_Fields functions 1025 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1028.

```
⟨ Define PICA_Fields functions 1026 ⟩ +≡
  CString PICA_Fields :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1029. Standard-SQL for Recordset. [LDF 2006.09.25.]

```
⟨ Declare PICA_Fields functions 1025 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1030.

```
⟨ Define PICA_Fields functions 1026 ⟩ +≡
  CString PICA_Fields :: GetDefaultSQL( )
  {
    return _T("[dbo].[PICA_Fields]");
  }
```

1031. Do Field Exchange. RFX-Unterstützung. [LDF 2006.09.25.]

```
⟨ Declare PICA_Fields functions 1025 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1032.

```
⟨ Define PICA_Fields functions 1026 ⟩ +≡
  void PICA_Fields::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[pica_field_id]"), m_pica_field_id);
    RFX_Text(pFX, _T("[pica_plus_field_code]"), m_pica_plus_field_code);
    RFX_Text(pFX, _T("[pica_3_field_code]"), m_pica_3_field_code);
    RFX_Text(pFX, _T("[description_german]"), m_description_german);
    RFX_Text(pFX, _T("[description_english]"), m_description_english);
  }
```

1033. Assert Valid. [LDF 2006.09.25.]

```
⟨ Declare PICA_Fields functions 1025 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1034.

```
⟨ Define PICA_Fields functions 1026 ⟩ +≡
#ifdef _DEBUG
  void PICA_Fields::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1035. Dump. [LDF 2006.09.25.]

```
⟨ Declare PICA_Fields functions 1025 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1036.

```
⟨ Define PICA_Fields functions 1026 ⟩ +≡
#ifdef _DEBUG
  void PICA_Fields::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1037. Putting PICA_Fields together.

1038. This is what gets written to `picaflds.h`.

```
⟨ picaflds.h 1038 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class PICA_Fields 1023 ⟩
```

1039. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ picaflds.web 1019 ⟩
⟨ Define PICA_Fields functions 1026 ⟩
```

1040. PICA_Categories_PICA_Fields (`pccatfld.web`).

Log

[LDF 2006.08.25.] Created this file. It contains code formerly in `pccatfld.h` and `pccatfld.cpp`.

```
⟨ pccatfld.web 1040 ⟩ ≡
  static char id_string[] = "$Id:␣pccatfld.web,v␣1.6␣2006/10/23␣14:23:01␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1060.

1041. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1042. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1043. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1044. PICA_Categories_PICA_Fields class declaration.

Log

[LDF 2006.08.25.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class PICA_Categories_PICA_Fields 1044 ⟩ ≡
  class PICA_Categories_PICA_Fields : public CRecordset {
  public: long m_pica_category_id;
    long m_pica_field_id;
    ⟨ Declare PICA_Categories_PICA_Fields functions 1046 ⟩
  };
```

This code is used in section 1059.

1045. Functions.

1046. Constructor.

```
⟨ Declare PICA_Categories_PICA_Fields functions 1046 ⟩ ≡
  PICA_Categories_PICA_Fields(CDatabase *pDatabase = NULL);
      DECLARE_DYNAMIC(PICA_Categories_PICA_Fields)
```

See also sections 1048, 1050, 1052, 1054, and 1056.

This code is used in section 1044.

1047.

⟨ Define **PICA_Categories_PICA_Fields** functions 1047 ⟩ ≡

```
  IMPLEMENT_DYNAMIC(PICA_Categories_PICA_Fields, CRecordset)
  PICA_Categories_PICA_Fields :: PICA_Categories_PICA_Fields(CDatabase *pdb)
  : CRecordset(pdb) {
    m_pica_category_id = 0;
    m_pica_field_id = 0;
    m_nFields = 2;
    m_nDefaultType = dynaset;
  }
```

See also sections 1049, 1051, 1053, 1055, and 1057.

This code is used in section 1060.

1048. Get default connect. [LDF 2006.08.25.]

⟨ Declare **PICA_Categories_PICA_Fields** functions 1046 ⟩ +≡

```
  virtual CString GetDefaultConnect( );
```

1049.

⟨ Define **PICA_Categories_PICA_Fields** functions 1047 ⟩ +≡

```
  CString PICA_Categories_PICA_Fields :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1050. Standard-SQL for Recordset. [LDF 2006.08.25.]

⟨ Declare **PICA_Categories_PICA_Fields** functions 1046 ⟩ +≡

```
  virtual CString GetDefaultSQL( );
```

1051.

⟨ Define **PICA_Categories_PICA_Fields** functions 1047 ⟩ +≡

```
  CString PICA_Categories_PICA_Fields :: GetDefaultSQL( )
  {
    return _T("[dbo].[PICA_Categories_PICA_Fields]");
  }
```

1052. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.25.]

⟨ Declare **PICA_Categories_PICA_Fields** functions 1046 ⟩ +≡

```
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1053.

```
⟨ Define PICA_Categories_PICA_Fields functions 1047 ⟩ +≡
  void PICA_Categories_PICA_Fields::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[pica_category_id]"), m_pica_category_id);
    RFX_Long(pFX, _T("[pica_field_id]"), m_pica_field_id);
  }
```

1054. Assert Valid. [LDF 2006.08.25.]

```
⟨ Declare PICA_Categories_PICA_Fields functions 1046 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1055.

```
⟨ Define PICA_Categories_PICA_Fields functions 1047 ⟩ +≡
#ifdef _DEBUG
  void PICA_Categories_PICA_Fields::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1056. Dump. [LDF 2006.08.25.]

```
⟨ Declare PICA_Categories_PICA_Fields functions 1046 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1057.

```
⟨ Define PICA_Categories_PICA_Fields functions 1047 ⟩ +≡
#ifdef _DEBUG
  void PICA_Categories_PICA_Fields::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1058. Putting PICA_Categories_PICA_Fields together.

1059. This is what gets written to `pccatfld.h`.

```
⟨ pccatfld.h 1059 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class PICA_Categories_PICA_Fields 1044 ⟩
```

1060. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ pccatfld.web 1040 ⟩
⟨ Define PICA_Categories_PICA_Fields functions 1047 ⟩
```

1061. Sources (sources.web).

Log

[LDF 2006.07.24.] Created this file. It contains code formerly in `sources.h` and `sources.cpp`.

```
⟨ sources.web 1061 ⟩ ≡
  static char id_string[] = "$Id:␣sources.web,v␣1.5␣2006/10/23␣14:25:49␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1081.

1062. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1063. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1064. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1065. Sources class declaration.

Log

[LDF 2006.07.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Sources 1065 ⟩ ≡
  class Sources : public CRecordset {
  public: long m_source_id;
    CStringA m_source_name;
    CStringA m_source_abbrev;
    CStringA m_source_address;
    ⟨ Declare Sources functions 1067 ⟩
  };
```

This code is used in section 1080.

1066. Functions.

1067. Constructor.

```
⟨ Declare Sources functions 1067 ⟩ ≡
  Sources(CDatabase *pDatabase = NULL); DECLARE_DYNAMIC(Sources)
```

See also sections 1069, 1071, 1073, 1075, and 1077.

This code is used in section 1065.

1068.

```
⟨ Define Sources functions 1068 ⟩ ≡
  IMPLEMENT_DYNAMIC(Sources, CRecordset)
  Sources :: Sources(CDatabase *pdb)
  : CRecordset(pdb) {
    m_source_id = 0;
    m_source_name = "";
    m_source_abbrev = "";
    m_source_address = "";
    m_nFields = 4;
    m_nDefaultType = dynaset;
  }
```

See also sections 1070, 1072, 1074, 1076, and 1078.

This code is used in section 1081.

1069. Get default connect. [LDF 2006.07.24.]

```
⟨ Declare Sources functions 1067 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1070.

```
⟨ Define Sources functions 1068 ⟩ +≡
  CString Sources :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1071. Standard-SQL for Recordset. [LDF 2006.07.24.]

```
⟨ Declare Sources functions 1067 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1072.

```
⟨ Define Sources functions 1068 ⟩ +≡
  CString Sources :: GetDefaultSQL( )
  {
    return _T("[dbo].[Sources]");
  }
```

1073. Do Field Exchange. RFX-Unterstützung. [LDF 2006.07.24.]

Log

[LDF 2006.09.11.] In the call to *RFX_Text* for *m_source_address*: Added a *nMaxLength* argument with value 1024.

```
⟨ Declare Sources functions 1067 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1074.

```
⟨ Define Sources functions 1068 ⟩ +≡
  void Sources::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[source_id]"), m_source_id);
    RFX_Text(pFX, _T("[source_name]"), m_source_name);
    RFX_Text(pFX, _T("[source_abbrev]"), m_source_abbrev);
    RFX_Text(pFX, _T("[source_address]"), m_source_address, 1024);
  }
```

1075. Assert Valid. [LDF 2006.07.24.]

```
⟨ Declare Sources functions 1067 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1076.

```
⟨ Define Sources functions 1068 ⟩ +≡
#ifdef _DEBUG
  void Sources::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1077. Dump. [LDF 2006.07.24.]

```
⟨ Declare Sources functions 1067 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1078.

```
⟨ Define Sources functions 1068 ⟩ +≡
#ifdef _DEBUG
  void Sources::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1079. Putting Sources together.

1080. This is what gets written to `sources.h`.

```
⟨ sources.h 1080 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Sources 1065 ⟩
```

1081. This is what gets compiled.

```
  ⟨ Include files 13 ⟩
  ⟨ sources.web 1061 ⟩
  ⟨ Define Sources functions 1068 ⟩
```

1082. Records (records.web).

Log

[LDF 2006.08.23.] Created this file. It contains code formerly in `records.h` and `records.cpp`.

```
⟨ records.web 1082 ⟩ ≡
  static char id_string[] = "$Id:␣records.web,v␣1.6␣2006/10/23␣14:25:17␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1102.

1083. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1084. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1085. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1086. Records class declaration.

Log

[LDF 2006.08.23.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records 1086 ⟩ ≡
  class Records : public CRecordset {
  public: long m_record_id;
    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;
    ⟨ Declare Records functions 1088 ⟩
  };
```

This code is used in section 1101.

1087. Functions.

1088. Constructor.

```
⟨ Declare Records functions 1088 ⟩ ≡
  Records(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records)
```

See also sections 1090, 1092, 1094, 1096, and 1098.

This code is used in section 1086.

1089.

```
⟨ Define Records functions 1089 ⟩ ≡
  IMPLEMENT_DYNAMIC(Records, CRecordset)
  Records::Records(CDatabase *pdb)
  : CRecordset(pdb) {
    m_record_id = 0;
    m_eln_original_entry = 0;
    m_eln_most_recent_change = 0;
    m_eln_status_change = 0;
    m_identification_number = "";
    m_date_original_entry;
    m_date_most_recent_change;
    m_date_status_change;
    m_source_id = 0;
    m_year_appearance_begin = 0;
    m_year_appearance_end = 0;
    m_year_appearance_rak_wb = 0;
    m_year_appearance_original = 0;
    m_nFields = 13;
    m_nDefaultType = dynaset;
  }
```

See also sections 1091, 1093, 1095, 1097, and 1099.

This code is used in section 1102.

1090. Get default connect. [LDF 2006.08.23.]

```
⟨ Declare Records functions 1088 ⟩ +≡
  virtual CString GetDefaultConnect();
```

1091.

```
⟨ Define Records functions 1089 ⟩ +≡
  CString Records::GetDefaultConnect()
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1092. Standard-SQL for Recordset. [LDF 2006.08.23.]

```
⟨ Declare Records functions 1088 ⟩ +≡
  virtual CString GetDefaultSQL();
```

1093.

```
⟨ Define Records functions 1089 ⟩ +≡
  CString Records :: GetDefaultSQL( )
  {
    return _T("[dbo].[Records]");
  }
```

1094. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.23.]

Log

[LDF 2006.09.11.] In the call to *RFX_Text* for *m_identification_number*: Added an *nMaxLength* argument with value 512.

```
⟨ Declare Records functions 1088 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1095.

```
⟨ Define Records functions 1089 ⟩ +≡
  void Records :: DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange :: outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Int(pFX, _T("[eln_original_entry]"), m_eln_original_entry);
    RFX_Int(pFX, _T("[eln_most_recent_change]"), m_eln_most_recent_change);
    RFX_Int(pFX, _T("[eln_status_change]"), m_eln_status_change);
    RFX_Text(pFX, _T("[identification_number]"), m_identification_number, 512);
    RFX_Date(pFX, _T("[date_original_entry]"), m_date_original_entry);
    RFX_Date(pFX, _T("[date_most_recent_change]"), m_date_most_recent_change);
    RFX_Date(pFX, _T("[date_status_change]"), m_date_status_change);
    RFX_Long(pFX, _T("[source_id]"), m_source_id);
    RFX_Int(pFX, _T("[year_appearance_begin]"), m_year_appearance_begin);
    RFX_Int(pFX, _T("[year_appearance_end]"), m_year_appearance_end);
    RFX_Int(pFX, _T("[year_appearance_rak_wb]"), m_year_appearance_rak_wb);
    RFX_Int(pFX, _T("[year_appearance_original]"), m_year_appearance_original);
  }
```

1096. Assert Valid. [LDF 2006.08.23.]

```
⟨ Declare Records functions 1088 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1097.

```
⟨ Define Records functions 1089 ⟩ +≡
#ifdef _DEBUG
  void Records::AssertValid() const
  {
    CRecordset::AssertValid();
  }
#endif
```

1098. Dump. [LDF 2006.08.23.]

```
⟨ Declare Records functions 1088 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1099.

```
⟨ Define Records functions 1089 ⟩ +≡
#ifdef _DEBUG
  void Records::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1100. Putting Records together.

1101. This is what gets written to `records.h`.

```
⟨ records.h 1101 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records 1086 ⟩
```

1102. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ records.web 1082 ⟩
⟨ Define Records functions 1089 ⟩
```

1103. Access_Numbers (accnums.web).

Log

[LDF 2006.08.24.] Created this file. It contains code formerly in `accnums.h` and `accnums.cpp`.

```
⟨ accnums.web 1103 ⟩ ≡
  static char id_string[] = "$Id:␣accnums.web,v␣1.7␣2006/10/23␣14:20:44␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1123.

1104. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1105. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1106. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1107. Access_Numbers class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Access_Numbers 1107 ⟩ ≡
  class Access_Numbers : public CRecordset {
  public: long m_access_number_id;
    CStringA m_access_number;
    long m_record_id;
    ⟨ Declare Access_Numbers functions 1109 ⟩
  };
```

This code is used in section 1122.

1108. Functions.

1109. Constructor.

```
⟨ Declare Access_Numbers functions 1109 ⟩ ≡
  Access_Numbers(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Access_Numbers)
```

See also sections 1111, 1113, 1115, 1117, and 1119.

This code is used in section 1107.

1110.

⟨ Define **Access_Numbers** functions 1110 ⟩ ≡
```
IMPLEMENT_DYNAMIC(Access_Numbers, CRecordset)
Access_Numbers::Access_Numbers(CDatabase *pdb)
: CRecordset(pdb) {
  m_access_number_id = 0;
  m_access_number = "";
  m_record_id = 0;
  m_nFields = 3;
  m_nDefaultType = dynaset;
}
```
See also sections 1112, 1114, 1116, 1118, and 1120.

This code is used in section 1123.

1111. Get default connect. [LDF 2006.08.24.]

⟨ Declare **Access_Numbers** functions 1109 ⟩ +≡
```
virtual CString GetDefaultConnect();
```

1112.

⟨ Define **Access_Numbers** functions 1110 ⟩ +≡
```
CString Access_Numbers::GetDefaultConnect()
{
  return _T(DATABASE_CONNECTION_STRING);
}
```

1113. Standard-SQL for Recordset. [LDF 2006.08.24.]

⟨ Declare **Access_Numbers** functions 1109 ⟩ +≡
```
virtual CString GetDefaultSQL();
```

1114.

⟨ Define **Access_Numbers** functions 1110 ⟩ +≡
```
CString Access_Numbers::GetDefaultSQL()
{
  return _T("[dbo].[Access_Numbers]");
}
```

1115. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

⟨ Declare **Access_Numbers** functions 1109 ⟩ +≡
```
virtual void DoFieldExchange(CFieldExchange *pFX);
```

1116.

```
⟨ Define Access_Numbers functions 1110 ⟩ +≡
  void Access_Numbers::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[access_number_id]"), m_access_number_id);
    RFX_Text(pFX, _T("[access_number]"), m_access_number);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
  }
```

1117. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Access_Numbers functions 1109 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1118.

```
⟨ Define Access_Numbers functions 1110 ⟩ +≡
#ifdef _DEBUG
  void Access_Numbers::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1119. Dump. [LDF 2006.08.24.]

```
⟨ Declare Access_Numbers functions 1109 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1120.

```
⟨ Define Access_Numbers functions 1110 ⟩ +≡
#ifdef _DEBUG
  void Access_Numbers::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1121. Putting Access_Numbers together.

1122. This is what gets written to accnums.h.

```
⟨ accnums.h 1122 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Access_Numbers 1107 ⟩
```

1123. This is what gets compiled.

```
  ⟨ Include files 13 ⟩
  ⟨ accnums.web 1103 ⟩
  ⟨ Define Access_Numbers functions 1110 ⟩
```

1124. Call_Numbers (callnums.web).

Log

[LDF 2006.08.24.] Created this file. It contains code formerly in `callnums.h` and `callnums.cpp`.

```
⟨ callnums.web 1124 ⟩ ≡
  static char id_string[] = "$Id:␣callnums.web,v␣1.6␣2006/10/23␣14:21:30␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1144.

1125. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1126. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1127. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1128. Call_Numbers class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Call_Numbers 1128 ⟩ ≡
  class Call_Numbers : public CRecordset {
  public: long m_call_number_id;
    CStringA m_call_number;
    int m_library_number;
    CStringA m_library_department;
    CStringA m_special_location;
    ⟨ Declare Call_Numbers functions 1130 ⟩
  };
```

This code is used in section 1143.

1129. Functions.

1130. Constructor.

```
⟨ Declare Call_Numbers functions 1130 ⟩ ≡
  Call_Numbers(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Call_Numbers)
```

See also sections 1132, 1134, 1136, 1138, and 1140.
This code is used in section 1128.

1131.

```
⟨ Define Call_Numbers functions 1131 ⟩ ≡
  IMPLEMENT_DYNAMIC(Call_Numbers, CRecordset)Call_Numbers::Call_Numbers(CDatabase
          *pdb)
  : CRecordset(pdb) {
    m_call_number_id = 0;
    m_call_number = "";
    m_library_number = 0;
    m_library_department = "";
    m_special_location = "";
    m_nFields = 5;
    m_nDefaultType = dynaset;
  }
```

See also sections 1133, 1135, 1137, 1139, and 1141.
This code is used in section 1144.

1132. Get default connect. [LDF 2006.08.24.]

```
⟨ Declare Call_Numbers functions 1130 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1133.

```
⟨ Define Call_Numbers functions 1131 ⟩ +≡
  CString Call_Numbers::GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1134. Standard-SQL for Recordset. [LDF 2006.08.24.]

```
⟨ Declare Call_Numbers functions 1130 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1135.

```
⟨ Define Call_Numbers functions 1131 ⟩ +≡
  CString Call_Numbers::GetDefaultSQL( )
  {
    return _T("[dbo].[Call_Numbers]");
  }
```

1136. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

```
⟨ Declare Call_Numbers functions 1130 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1137.

```
⟨ Define Call_Numbers functions 1131 ⟩ +≡
  void Call_Numbers::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[call_number_id]"), m_call_number_id);
    RFX_Text(pFX, _T("[call_number]"), m_call_number);
    RFX_Int(pFX, _T("[library_number]"), m_library_number);
    RFX_Text(pFX, _T("[library_department]"), m_library_department);
    RFX_Text(pFX, _T("[special_location]"), m_special_location);
  }
```

1138. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Call_Numbers functions 1130 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1139.

```
⟨ Define Call_Numbers functions 1131 ⟩ +≡
#ifdef _DEBUG
  void Call_Numbers::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1140. Dump. [LDF 2006.08.24.]

```
⟨ Declare Call_Numbers functions 1130 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1141.

```
⟨ Define Call_Numbers functions 1131 ⟩ +≡
#ifdef _DEBUG
  void Call_Numbers::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1142. Putting Call_Numbers together.

1143. This is what gets written to `callnums.h`.

```
⟨ callnums.h 1143 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Call_Numbers 1128 ⟩
```

1144. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ callnums.web 1124 ⟩
⟨ Define Call_Numbers functions 1131 ⟩
```

1145. Records_Call_Numbers (rccllnms.web).

Log

[LDF 2006.08.24.] Created this file. It contains code formerly in rccllnms.h and rccllnms.cpp.

```
⟨ rccllnms.web 1145 ⟩ ≡
  static char id_string[] = "$Id:␣rccllnms.web,v␣1.5␣2006/10/23␣14:24:02␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1165.

1146. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1147. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1148. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1149. Records_Call_Numbers class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records_Call_Numbers 1149 ⟩ ≡
  class Records_Call_Numbers : public CRecordset {
  public: long m_record_id;
    long m_call_number_id;
    ⟨ Declare Records_Call_Numbers functions 1151 ⟩
  };
```

This code is used in section 1164.

1150. Functions.

1151. Constructor.

```
⟨ Declare Records_Call_Numbers functions 1151 ⟩ ≡
  Records_Call_Numbers(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records_Call_Numbers)
```

See also sections 1153, 1155, 1157, 1159, and 1161.

This code is used in section 1149.

1152.

⟨ Define **Records_Call_Numbers** functions 1152 ⟩ ≡

```
IMPLEMENT_DYNAMIC(Records_Call_Numbers,
        CRecordset)Records_Call_Numbers::Records_Call_Numbers(CDatabase *pdb)
  : CRecordset(pdb) {
    m_record_id = 0;
    m_call_number_id = 0;
    m_nFields = 2;
    m_nDefaultType = dynaset;
  }
```

See also sections 1154, 1156, 1158, 1160, and 1162.

This code is used in section 1165.

1153. Get default connect. [LDF 2006.08.24.]

⟨ Declare **Records_Call_Numbers** functions 1151 ⟩ +≡

```
virtual CString GetDefaultConnect();
```

1154.

⟨ Define **Records_Call_Numbers** functions 1152 ⟩ +≡

```
CString Records_Call_Numbers::GetDefaultConnect()
{
  return _T(DATABASE_CONNECTION_STRING);
}
```

1155. Standard-SQL for Recordset. [LDF 2006.08.24.]

⟨ Declare **Records_Call_Numbers** functions 1151 ⟩ +≡

```
virtual CString GetDefaultSQL();
```

1156.

⟨ Define **Records_Call_Numbers** functions 1152 ⟩ +≡

```
CString Records_Call_Numbers::GetDefaultSQL()
{
  return _T("[dbo].[Records_Call_Numbers]");
}
```

1157. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

⟨ Declare **Records_Call_Numbers** functions 1151 ⟩ +≡

```
virtual void DoFieldExchange(CFieldExchange *pFX);
```

1158.

```
⟨ Define Records_Call_Numbers functions 1152 ⟩ +≡
  void Records_Call_Numbers::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[call_number_id]"), m_call_number_id);
  }
```

1159. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Records_Call_Numbers functions 1151 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1160.

```
⟨ Define Records_Call_Numbers functions 1152 ⟩ +≡
#ifdef _DEBUG
  void Records_Call_Numbers::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1161. Dump. [LDF 2006.08.24.]

```
⟨ Declare Records_Call_Numbers functions 1151 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1162.

```
⟨ Define Records_Call_Numbers functions 1152 ⟩ +≡
#ifdef _DEBUG
  void Records_Call_Numbers::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1163. Putting Records_Call_Numbers together.

1164. This is what gets written to `rccllnms.h`.

```
⟨ rccllnms.h 1164 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records_Call_Numbers 1149 ⟩
```

1165. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ rccllnms.web 1145 ⟩
⟨ Define Records_Call_Numbers functions 1152 ⟩
```

1166. Exemplar_Production_Numbers (exprnums.web).

Log

[LDF 2006.08.24.] Created this file. It contains code formerly in `exprnums.h` and `exprnums.cpp`.

```
⟨ exprnums.web 1166 ⟩ ≡
  static char id_string[] = "$Id:␣exprnums.web,v␣1.6␣2006/10/23␣14:22:16␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1186.

1167. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1168. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1169. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1170. Exemplar_Production_Numbers class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

[LDF 2006.08.29.] Changed the declaration of **class Exemplar_Production_Numbers** to correspond to changes made in the **Exemplar_Production_Numbers** database table. Replaced the variable *m_exemplar_production_n* with *m_exemplar_production_number_numeric* and *m_exemplar_production_number_text*. Changed other code accordingly.

```
⟨ Declare class Exemplar_Production_Numbers 1170 ⟩ ≡
  class Exemplar_Production_Numbers : public CRecordset {
  public: long m_exemplar_production_number_id;
    LONGLONG m_exemplar_production_number_numeric;
    CStringA m_exemplar_production_number_text;
    long m_record_id;
    ⟨ Declare Exemplar_Production_Numbers functions 1172 ⟩
  };
```

This code is used in section 1185.

1171. Functions.

1172. Constructor.

```
⟨ Declare Exemplar_Production_Numbers functions 1172 ⟩ ≡
  Exemplar_Production_Numbers(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Exemplar_Production_Numbers)
```

See also sections 1174, 1176, 1178, 1180, and 1182.

This code is used in section 1170.

1173.

```
⟨ Define Exemplar_Production_Numbers functions 1173 ⟩ ≡
  IMPLEMENT_DYNAMIC(Exemplar_Production_Numbers, CRecordset)
  Exemplar_Production_Numbers::Exemplar_Production_Numbers(CDatabase *pdb)
  : CRecordset(pdb) {
    m_exemplar_production_number_id = 0;
    m_exemplar_production_number_numeric = 0;
    m_exemplar_production_number_text = "";
    m_record_id = 0;
    m_nFields = 4;
    m_nDefaultType = dynaset;
  }
```

See also sections 1175, 1177, 1179, 1181, and 1183.

This code is used in section 1186.

1174. Get default connect. [LDF 2006.08.24.]

```
⟨ Declare Exemplar_Production_Numbers functions 1172 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1175.

```
⟨ Define Exemplar_Production_Numbers functions 1173 ⟩ +≡
  CString Exemplar_Production_Numbers::GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1176. Standard-SQL for Recordset. [LDF 2006.08.24.]

```
⟨ Declare Exemplar_Production_Numbers functions 1172 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1177.

```
⟨ Define Exemplar_Production_Numbers functions 1173 ⟩ +≡
  CString Exemplar_Production_Numbers::GetDefaultSQL( )
  {
    return _T("[dbo].[Exemplar_Production_Numbers]");
  }
```

1178. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

```
⟨ Declare Exemplar_Production_Numbers functions 1172 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1179.

⟨ Define **Exemplar_Production_Numbers** functions 1173 ⟩ +≡

```
  void Exemplar_Production_Numbers::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[exemplar_production_number_id]"), m_exemplar_production_number_id);
    RFX_BigInt(pFX, _T("[exemplar_production_number_numeric]"),
        m_exemplar_production_number_numeric);
    RFX_Text(pFX, _T("[exemplar_production_number_text]"), m_exemplar_production_number_text);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
  }
```

1180. Assert Valid. [LDF 2006.08.24.]

⟨ Declare **Exemplar_Production_Numbers** functions 1172 ⟩ +≡

```
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1181.

⟨ Define **Exemplar_Production_Numbers** functions 1173 ⟩ +≡

```
#ifdef _DEBUG
  void Exemplar_Production_Numbers::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1182. Dump. [LDF 2006.08.24.]

⟨ Declare **Exemplar_Production_Numbers** functions 1172 ⟩ +≡

```
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1183.

⟨ Define **Exemplar_Production_Numbers** functions 1173 ⟩ +≡

```
#ifdef _DEBUG
  void Exemplar_Production_Numbers::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1184. Putting Exemplar_Production_Numbers together.

1185. This is what gets written to `exprnums.h`.

⟨ `exprnums.h` 1185 ⟩ ≡

```
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Exemplar_Production_Numbers 1170 ⟩
```

1186. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ exprnums.web 1166 ⟩
⟨ Define Exemplar_Production_Numbers functions 1173 ⟩
```

1187. Bibliographic_Type_Codes (bbtpcds.web).

Log

[LDF 2006.09.06.] Created this file. It contains code formerly in `bbtpcds.h` and `bbtpcds.cpp`.

```
⟨ bbtpcds.web 1187 ⟩ ≡
  static char id_string[] = "$Id:␣bbtpcds.web,v␣1.7␣2006/10/23␣14:21:13␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1207.

1188. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1189. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1190. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1191. Bibliographic_Type_Codes class declaration.

Log

[LDF 2006.09.06.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Bibliographic_Type_Codes 1191 ⟩ ≡
  class Bibliographic_Type_Codes : public CRecordset {
  public: long m_bibliographic_type_code_id;
    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;
    ⟨ Declare Bibliographic_Type_Codes functions 1193 ⟩
  };
```

This code is used in section 1206.

1192. Functions.

1193. Constructor.

```
⟨ Declare Bibliographic_Type_Codes functions 1193 ⟩ ≡
  Bibliographic_Type_Codes(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Bibliographic_Type_Codes)
```

See also sections 1195, 1197, 1199, 1201, and 1203.

This code is used in section 1191.

1194.

```
⟨ Define Bibliographic_Type_Codes functions 1194 ⟩ ≡
  IMPLEMENT_DYNAMIC(Bibliographic_Type_Codes, CRecordset)
  Bibliographic_Type_Codes::Bibliographic_Type_Codes(CDatabase *pdb)
  : CRecordset(pdb) {
    m_bibliographic_type_code_id = 0;
    m_physical_form_code = "";
    m_physical_form_material_name_english = "";
    m_physical_form_material_name_german = "";
    m_bibliographic_representation_code = "";
    m_bibliographic_representation_description_english = "";
    m_bibliographic_representation_description_german = "";
    m_description_status_code = "";
    m_description_status_description_english = "";
    m_description_status_description_german = "";
    m_nFields = 10;
    m_nDefaultType = dynaset;
  }
```

See also sections 1196, 1198, 1200, 1202, and 1204.

This code is used in section 1207.

1195. Get default connect. [LDF 2006.09.06.]

```
⟨ Declare Bibliographic_Type_Codes functions 1193 ⟩ +≡
  virtual CString GetDefaultConnect();
```

1196.

```
⟨ Define Bibliographic_Type_Codes functions 1194 ⟩ +≡
  CString Bibliographic_Type_Codes::GetDefaultConnect()
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1197. Standard-SQL for Recordset. [LDF 2006.09.06.]

```
⟨ Declare Bibliographic_Type_Codes functions 1193 ⟩ +≡
  virtual CString GetDefaultSQL();
```

1198.

```
⟨ Define Bibliographic_Type_Codes functions 1194 ⟩ +≡
  CString Bibliographic_Type_Codes::GetDefaultSQL( )
  {
    return _T("[dbo].[Bibliographic_Type_Codes]");
  }
```

1199. Do Field Exchange. RFX-Unterstützung. [LDF 2006.09.06.]

```
⟨ Declare Bibliographic_Type_Codes functions 1193 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1200.

```
⟨ Define Bibliographic_Type_Codes functions 1194 ⟩ +≡
  void Bibliographic_Type_Codes::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[bibliographic_type_code_id]"), m_bibliographic_type_code_id);
    RFX_Text(pFX, _T("[physical_form_code]"), m_physical_form_code);
    RFX_Text(pFX, _T("[physical_form_material_name_english]"),
        m_physical_form_material_name_english);
    RFX_Text(pFX, _T("[physical_form_material_name_german]"),
        m_physical_form_material_name_german);
    RFX_Text(pFX, _T("[bibliographic_representation_code]"), m_bibliographic_representation_code);
    RFX_Text(pFX, _T("[bibliographic_representation_description_english]"),
        m_bibliographic_representation_description_english);
    RFX_Text(pFX, _T("[bibliographic_representation_description_german]"),
        m_bibliographic_representation_description_german);
    RFX_Text(pFX, _T("[description_status_code]"), m_description_status_code);
    RFX_Text(pFX, _T("[description_status_description_english]"),
        m_description_status_description_english);
    RFX_Text(pFX, _T("[description_status_description_german]"),
        m_description_status_description_german);
  }
```

1201. Assert Valid. [LDF 2006.09.06.]

```
⟨ Declare Bibliographic_Type_Codes functions 1193 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1202.

```
⟨ Define Bibliographic_Type_Codes functions 1194 ⟩ +≡
#ifdef _DEBUG
  void Bibliographic_Type_Codes::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1203. Dump. [LDF 2006.09.06.]

```
⟨ Declare Bibliographic_Type_Codes functions 1193 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1204.

```
⟨ Define Bibliographic_Type_Codes functions 1194 ⟩ +≡
#ifdef _DEBUG
  void Bibliographic_Type_Codes::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1205. Putting Bibliographic_Type_Codes together.

1206. This is what gets written to `bbtpcds.h`.

```
⟨ bbtpcds.h 1206 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Bibliographic_Type_Codes 1191 ⟩
```

1207. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ bbtpcds.web 1187 ⟩
⟨ Define Bibliographic_Type_Codes functions 1194 ⟩
```

1208. Bibliographic_Types (bibtyps.web).

Log

[LDF 2006.08.24.] Created this file. It contains code formerly in bibtyps.h and bibtyps.cpp.

```
⟨ bibtyps.web 1208 ⟩ ≡
  static char id_string[] = "$Id:␣bibtyps.web,v␣1.7␣2006/10/23␣14:21:22␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1228.

1209. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1210. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1211. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1212. Bibliographic_Types class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Bibliographic_Types 1212 ⟩ ≡
  class Bibliographic_Types : public CRecordset {
  public: ⟨ Declare Bibliographic_Types functions 1214 ⟩
    long m_bibliographic_type_id;
    CStringA m_physical_form;
    CStringA m_bibliographic_representation;
    CStringA m_description_status;
    CStringA m_miscellaneous;
    CStringA m_bibliographic_representation_refinement;
    CStringA m_transliteration_code;
  };
```

This code is used in section 1227.

1213. Functions.

1214. Constructor.

```
⟨ Declare Bibliographic_Types functions 1214 ⟩ ≡
```

```
  Bibliographic_Types(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Bibliographic_Types)
```

See also sections 1216, 1218, 1220, 1222, and 1224.

This code is used in section 1212.

1215.

```
⟨ Define Bibliographic_Types functions 1215 ⟩ ≡
  IMPLEMENT_DYNAMIC(Bibliographic_Types, CRecordset)
  Bibliographic_Types :: Bibliographic_Types(CDatabase *pdb)
  : CRecordset(pdb) {
    m_bibliographic_type_id = 0;
    m_physical_form = "";
    m_bibliographic_representation = "";
    m_description_status = "";
    m_miscellaneous = "";
    m_bibliographic_representation_refinement = "";
    m_transliteration_code = "";
    m_nFields = 7;
    m_nDefaultType = dynaset;
  }
```

See also sections 1217, 1219, 1221, 1223, and 1225.

This code is used in section 1228.

1216. Get default connect. [LDF 2006.08.24.]

```
⟨ Declare Bibliographic_Types functions 1214 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1217.

```
⟨ Define Bibliographic_Types functions 1215 ⟩ +≡
  CString Bibliographic_Types :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1218. Standard-SQL for Recordset. [LDF 2006.08.24.]

```
⟨ Declare Bibliographic_Types functions 1214 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1219.

```
⟨ Define Bibliographic_Types functions 1215 ⟩ +≡
  CString Bibliographic_Types :: GetDefaultSQL( )
  {
    return _T("[dbo].[Bibliographic_Types]");
  }
```

1220. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

```
⟨ Declare Bibliographic_Types functions 1214 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1221.

```
⟨ Define Bibliographic_Types functions 1215 ⟩ +≡
  void Bibliographic_Types::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[bibliographic_type_id]"), m_bibliographic_type_id);
    RFX_Text(pFX, _T("[physical_form]"), m_physical_form);
    RFX_Text(pFX, _T("[bibliographic_representation]"), m_bibliographic_representation);
    RFX_Text(pFX, _T("[description_status]"), m_description_status);
    RFX_Text(pFX, _T("[miscellaneous]"), m_miscellaneous);
    RFX_Text(pFX, _T("[bibliographic_representation_refinement]"),
         m_bibliographic_representation_refinement);
    RFX_Text(pFX, _T("[transliteration_code]"), m_transliteration_code);
  }
```

1222. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Bibliographic_Types functions 1214 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1223.

```
⟨ Define Bibliographic_Types functions 1215 ⟩ +≡
#ifdef _DEBUG
  void Bibliographic_Types::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1224. Dump. [LDF 2006.08.24.]

```
⟨ Declare Bibliographic_Types functions 1214 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1225.

```
⟨ Define Bibliographic_Types functions 1215 ⟩ +≡
#ifdef _DEBUG
  void Bibliographic_Types::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1226. Putting Bibliographic_Types together.

1227. This is what gets written to `bibtyps.h`.

⟨ `bibtyps.h` 1227 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ **using** declarations for namespaces 191 ⟩
  ⟨ Declare **class Bibliographic_Types** 1212 ⟩

1228. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ bibtyps.web 1208 ⟩
⟨ Define Bibliographic_Types functions 1215 ⟩
```

1229. Records_Bibliographic_Types (rcbbtyps.web).

Log

[LDF 2006.08.24.] Created this file. It contains code formerly in `rcbbtyps.h` and `rcbbtyps.cpp`.

```
⟨ rcbbtyps.web 1229 ⟩ ≡
  static char id_string[] = "$Id:␣rcbbtyps.web,v␣1.5␣2006/10/23␣14:23:54␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1249.

1230. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1231. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1232. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1233. Records_Bibliographic_Types class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records_Bibliographic_Types 1233 ⟩ ≡
  class Records_Bibliographic_Types : public CRecordset {
  public: long m_record_id;
    long m_bibliographic_type_id;
    ⟨ Declare Records_Bibliographic_Types functions 1235 ⟩
  };
```

This code is used in section 1248.

1234. Functions.

1235. Constructor.

```
⟨ Declare Records_Bibliographic_Types functions 1235 ⟩ ≡
  Records_Bibliographic_Types(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records_Bibliographic_Types)
```

See also sections 1237, 1239, 1241, 1243, and 1245.

This code is used in section 1233.

1236.

⟨ Define **Records_Bibliographic_Types** functions 1236 ⟩ ≡

```
  IMPLEMENT_DYNAMIC(Records_Bibliographic_Types, CRecordset)
  Records_Bibliographic_Types::Records_Bibliographic_Types(CDatabase *pdb)
  : CRecordset(pdb) {
    m_record_id = 0;
    m_bibliographic_type_id = 0;
    m_nFields = 2;
    m_nDefaultType = dynaset;
  }
```

See also sections 1238, 1240, 1242, 1244, and 1246.

This code is used in section 1249.

1237. Get default connect. [LDF 2006.08.24.]

⟨ Declare **Records_Bibliographic_Types** functions 1235 ⟩ +≡

```
  virtual CString GetDefaultConnect();
```

1238.

⟨ Define **Records_Bibliographic_Types** functions 1236 ⟩ +≡

```
  CString Records_Bibliographic_Types::GetDefaultConnect()
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1239. Standard-SQL for Recordset. [LDF 2006.08.24.]

⟨ Declare **Records_Bibliographic_Types** functions 1235 ⟩ +≡

```
  virtual CString GetDefaultSQL();
```

1240.

⟨ Define **Records_Bibliographic_Types** functions 1236 ⟩ +≡

```
  CString Records_Bibliographic_Types::GetDefaultSQL()
  {
    return _T("[dbo].[Records_Bibliographic_Types]");
  }
```

1241. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

⟨ Declare **Records_Bibliographic_Types** functions 1235 ⟩ +≡

```
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1242.

```
⟨ Define Records_Bibliographic_Types functions 1236 ⟩ +≡
  void Records_Bibliographic_Types::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX->SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[bibliographic_type_id]"), m_bibliographic_type_id);
  }
```

1243. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Records_Bibliographic_Types functions 1235 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid() const;
#endif
```

1244.

```
⟨ Define Records_Bibliographic_Types functions 1236 ⟩ +≡
#ifdef _DEBUG
  void Records_Bibliographic_Types::AssertValid() const
  {
    CRecordset::AssertValid();
  }
#endif
```

1245. Dump. [LDF 2006.08.24.]

```
⟨ Declare Records_Bibliographic_Types functions 1235 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1246.

```
⟨ Define Records_Bibliographic_Types functions 1236 ⟩ +≡
#ifdef _DEBUG
  void Records_Bibliographic_Types::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1247. Putting Records_Bibliographic_Types together.

1248. This is what gets written to `rcbbtyps.h`.

```
⟨ rcbbtyps.h 1248 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records_Bibliographic_Types 1233 ⟩
```

1249. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ rcbbtyps.web 1229 ⟩
⟨ Define Records_Bibliographic_Types functions 1236 ⟩
```

1250. Physical_Descriptions (physdesc.web).

Log

[LDF 2006.08.25.] Created this file. It contains code formerly in `physdesc.h` and `physdesc.cpp`.

```
⟨ physdesc.web 1250 ⟩ ≡
  static char id_string[] = "$Id:␣physdesc.web,v␣1.6␣2006/10/23␣14:23:09␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.
This code is used in section 1270.

1251. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1252. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1253. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1254. Physical_Descriptions class declaration.

Log

[LDF 2006.08.25.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Physical_Descriptions 1254 ⟩ ≡
  class Physical_Descriptions : public CRecordset {
  public: long m_physical_description_id;
    CStringA m_text;
    long m_pica_category_id;
    long m_pica_field_id;
    ⟨ Declare Physical_Descriptions functions 1256 ⟩
  };
```

This code is used in section 1269.

1255. Functions.

1256. Constructor.

```
⟨ Declare Physical_Descriptions functions 1256 ⟩ ≡
  Physical_Descriptions(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Physical_Descriptions)
```

See also sections 1258, 1260, 1262, 1264, and 1266.
This code is used in section 1254.

1257.

```
⟨ Define Physical_Descriptions functions 1257 ⟩ ≡
  IMPLEMENT_DYNAMIC(Physical_Descriptions, CRecordset)
  Physical_Descriptions::Physical_Descriptions(CDatabase *pdb)
  : CRecordset(pdb) {
    m_physical_description_id = 0;
    m_text = "";
    m_pica_category_id = 0;
    m_pica_field_id = 0;
    m_nFields = 4;
    m_nDefaultType = dynaset;
  }
```

See also sections 1259, 1261, 1263, 1265, and 1267.

This code is used in section 1270.

1258. Get default connect. [LDF 2006.08.25.]

```
⟨ Declare Physical_Descriptions functions 1256 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1259.

```
⟨ Define Physical_Descriptions functions 1257 ⟩ +≡
  CString Physical_Descriptions::GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1260. Standard-SQL for Recordset. [LDF 2006.08.25.]

```
⟨ Declare Physical_Descriptions functions 1256 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1261.

```
⟨ Define Physical_Descriptions functions 1257 ⟩ +≡
  CString Physical_Descriptions::GetDefaultSQL( )
  {
    return _T("[dbo].[Physical_Descriptions]");
  }
```

1262. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.25.]

```
⟨ Declare Physical_Descriptions functions 1256 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1263.

```
⟨ Define Physical_Descriptions functions 1257 ⟩ +≡
  void Physical_Descriptions::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[physical_description_id]"), m_physical_description_id);
    RFX_Text(pFX, _T("[text]"), m_text, 256);
    RFX_Long(pFX, _T("[pica_category_id]"), m_pica_category_id);
    RFX_Long(pFX, _T("[pica_field_id]"), m_pica_field_id);
  }
```

1264. Assert Valid. [LDF 2006.08.25.]

```
⟨ Declare Physical_Descriptions functions 1256 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1265.

```
⟨ Define Physical_Descriptions functions 1257 ⟩ +≡
#ifdef _DEBUG
  void Physical_Descriptions::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1266. Dump. [LDF 2006.08.25.]

```
⟨ Declare Physical_Descriptions functions 1256 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1267.

```
⟨ Define Physical_Descriptions functions 1257 ⟩ +≡
#ifdef _DEBUG
  void Physical_Descriptions::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1268. Putting Physical_Descriptions together.

1269. This is what gets written to `physdesc.h`.

```
⟨ physdesc.h 1269 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Physical_Descriptions 1254 ⟩
```

1270. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ physdesc.web 1250 ⟩
⟨ Define Physical_Descriptions functions 1257 ⟩
```

1271. Records_Physical_Descriptions (rcphsdsc.web).

Log

[LDF 2006.08.25.] Created this file. It contains code formerly in rcphsdsc.h and rcphsdsc.cpp.

```
⟨ rcphsdsc.web 1271 ⟩ ≡
  static char id_string[] = "$Id:␣rcphsdsc.web,v␣1.5␣2006/10/23␣14:24:32␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1291.

1272. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1273. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1274. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1275. Records_Physical_Descriptions class declaration.

Log

[LDF 2006.08.25.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records_Physical_Descriptions 1275 ⟩ ≡
  class Records_Physical_Descriptions : public CRecordset {
  public: long m_record_id;
    long m_physical_description_id;
    ⟨ Declare Records_Physical_Descriptions functions 1277 ⟩
  };
```

This code is used in section 1290.

1276. Functions.

1277. Constructor.

```
⟨ Declare Records_Physical_Descriptions functions 1277 ⟩ ≡
  Records_Physical_Descriptions(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records_Physical_Descriptions)
```

See also sections 1279, 1281, 1283, 1285, and 1287.

This code is used in section 1275.

1278.

```
⟨ Define Records_Physical_Descriptions functions 1278 ⟩ ≡
  IMPLEMENT_DYNAMIC(Records_Physical_Descriptions, CRecordset)
  Records_Physical_Descriptions :: Records_Physical_Descriptions(CDatabase *pdb)
  : CRecordset(pdb) {
    m_record_id = 0;
    m_physical_description_id = 0;
    m_nFields = 2;
    m_nDefaultType = dynaset;
  }
```

See also sections 1280, 1282, 1284, 1286, and 1288.

This code is used in section 1291.

1279. Get default connect. [LDF 2006.08.25.]

```
⟨ Declare Records_Physical_Descriptions functions 1277 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1280.

```
⟨ Define Records_Physical_Descriptions functions 1278 ⟩ +≡
  CString Records_Physical_Descriptions :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1281. Standard-SQL for Recordset. [LDF 2006.08.25.]

```
⟨ Declare Records_Physical_Descriptions functions 1277 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1282.

```
⟨ Define Records_Physical_Descriptions functions 1278 ⟩ +≡
  CString Records_Physical_Descriptions :: GetDefaultSQL( )
  {
    return _T("[dbo].[Records_Physical_Descriptions]");
  }
```

1283. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.25.]

```
⟨ Declare Records_Physical_Descriptions functions 1277 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1284.

```
⟨ Define Records_Physical_Descriptions functions 1278 ⟩ +≡
  void Records_Physical_Descriptions::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[physical_description_id]"), m_physical_description_id);
  }
```

1285. Assert Valid. [LDF 2006.08.25.]

```
⟨ Declare Records_Physical_Descriptions functions 1277 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1286.

```
⟨ Define Records_Physical_Descriptions functions 1278 ⟩ +≡
#ifdef _DEBUG
  void Records_Physical_Descriptions::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1287. Dump. [LDF 2006.08.25.]

```
⟨ Declare Records_Physical_Descriptions functions 1277 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1288.

```
⟨ Define Records_Physical_Descriptions functions 1278 ⟩ +≡
#ifdef _DEBUG
  void Records_Physical_Descriptions::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1289. Putting Records_Physical_Descriptions together.

1290. This is what gets written to `rcphsdsc.h`.

```
⟨ rcphsdsc.h 1290 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records_Physical_Descriptions 1275 ⟩
```

1291. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ rcphsdsc.web 1271 ⟩
⟨ Define Records_Physical_Descriptions functions 1278 ⟩
```

1292. Authors (authors.web).

Log

[LDF 2006.08.24.] Created this file. It contains code formerly in `authors.h` and `authors.cpp`.

```
⟨ authors.web 1292 ⟩ ≡
  static char id_string[] = "$Id:␣authors.web,v␣1.7␣2006/10/23␣14:21:03␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1312.

1293. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1294. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1295. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1296. Authors class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Authors 1296 ⟩ ≡
  class Authors : public CRecordset {
  public: long m_author_id;
    CStringA m_given_name;
    CStringA m_surname;
    CStringA m_prefix;
    LONGLONGm_id_number_ppn;
    ⟨ Declare Authors functions 1298 ⟩
  };
```

This code is used in section 1311.

1297. Functions.

1298. Constructor.

```
⟨ Declare Authors functions 1298 ⟩ ≡
  Authors(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Authors)
```

See also sections 1300, 1302, 1304, 1306, and 1308.
This code is used in section 1296.

1299.

```
⟨ Define Authors functions 1299 ⟩ ≡
  IMPLEMENT_DYNAMIC(Authors, CRecordset)
  Authors::Authors(CDatabase *pdb)
  : CRecordset(pdb) {
    m_author_id = 0;
    m_given_name = "";
    m_surname = "";
    m_prefix = "";
    m_id_number_ppn = 0;
    m_nFields = 5;
    m_nDefaultType = dynaset;
  }
```

See also sections 1301, 1303, 1305, 1307, and 1309.
This code is used in section 1312.

1300. Get default connect. [LDF 2006.08.24.]

```
⟨ Declare Authors functions 1298 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1301.

```
⟨ Define Authors functions 1299 ⟩ +≡
  CString Authors::GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1302. Standard-SQL for Recordset. [LDF 2006.08.24.]

```
⟨ Declare Authors functions 1298 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1303.

```
⟨ Define Authors functions 1299 ⟩ +≡
  CString Authors::GetDefaultSQL( )
  {
    return _T("[dbo].[Authors]");
  }
```

1304. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

```
⟨ Declare Authors functions 1298 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1305.

```
⟨ Define Authors functions 1299 ⟩ +≡
  void Authors :: DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange :: outputColumn);
    RFX_Long(pFX, _T("[author_id]"), m_author_id);
    RFX_Text(pFX, _T("[given_name]"), m_given_name);
    RFX_Text(pFX, _T("[surname]"), m_surname);
    RFX_Text(pFX, _T("[prefix]"), m_prefix);
    RFX_BigInt(pFX, _T("[id_number_ppn]"), m_id_number_ppn);
  }
```

1306. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Authors functions 1298 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1307.

```
⟨ Define Authors functions 1299 ⟩ +≡
#ifdef _DEBUG
  void Authors :: AssertValid( ) const
  {
    CRecordset :: AssertValid( );
  }
#endif
```

1308. Dump. [LDF 2006.08.24.]

```
⟨ Declare Authors functions 1298 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1309.

```
⟨ Define Authors functions 1299 ⟩ +≡
#ifdef _DEBUG
  void Authors :: Dump(CDumpContext &dc) const
  {
    CRecordset :: Dump(dc);
  }
#endif
```

1310. Putting Authors together.

1311. This is what gets written to `authors.h`.

```
⟨ authors.h 1311 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Authors 1296 ⟩
```

1312. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ authors.web 1292 ⟩
⟨ Define Authors functions 1299 ⟩
```

1313. Records_Authors (recathrs.web).

Log

[LDF 2006.09.26.] Created this file. It contains code formerly in Records_Authors.h and Records_Authors.cpp.

```
⟨ recathrs.web 1313 ⟩ ≡
  static char id_string[] = "$Id:␣recathrs.web,v␣1.5␣2006/10/23␣14:24:46␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1333.

1314. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1315. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1316. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1317. Records_Authors class declaration.

Log

[LDF 2006.08.25.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records_Authors 1317 ⟩ ≡
  class Records_Authors : public CRecordset {
  public: long m_record_id;
    long m_author_id;
    ⟨ Declare Records_Authors functions 1319 ⟩
  };
```

This code is used in section 1332.

1318. Functions.

1319. Constructor.

```
⟨ Declare Records_Authors functions 1319 ⟩ ≡
  Records_Authors(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records_Authors)
```

See also sections 1321, 1323, 1325, 1327, and 1329.

This code is used in section 1317.

1320.

```
⟨ Define Records_Authors functions 1320 ⟩ ≡
  IMPLEMENT_DYNAMIC(Records_Authors, CRecordset)
  Records_Authors :: Records_Authors(CDatabase *pdb)
  : CRecordset(pdb) {
    m_record_id = 0;
    m_author_id = 0;
    m_nFields = 2;
    m_nDefaultType = dynaset;
  }
```

See also sections 1322, 1324, 1326, 1328, and 1330.

This code is used in section 1333.

1321. Get default connect. [LDF 2006.08.25.]

```
⟨ Declare Records_Authors functions 1319 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1322.

```
⟨ Define Records_Authors functions 1320 ⟩ +≡
  CString Records_Authors :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1323. Standard-SQL for Recordset. [LDF 2006.08.25.]

```
⟨ Declare Records_Authors functions 1319 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1324.

```
⟨ Define Records_Authors functions 1320 ⟩ +≡
  CString Records_Authors :: GetDefaultSQL( )
  {
    return _T("[dbo].[Records_Authors]");
  }
```

1325. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.25.]

```
⟨ Declare Records_Authors functions 1319 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1326.

```
⟨ Define Records_Authors functions 1320 ⟩ +≡
  void Records_Authors::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[author_id]"), m_author_id);
  }
```

1327. Assert Valid. [LDF 2006.08.25.]

```
⟨ Declare Records_Authors functions 1319 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1328.

```
⟨ Define Records_Authors functions 1320 ⟩ +≡
#ifdef _DEBUG
  void Records_Authors::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1329. Dump. [LDF 2006.08.25.]

```
⟨ Declare Records_Authors functions 1319 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1330.

```
⟨ Define Records_Authors functions 1320 ⟩ +≡
#ifdef _DEBUG
  void Records_Authors::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1331. Putting Records_Authors together.

1332. This is what gets written to `recathrs.h`.

```
⟨ recathrs.h 1332 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records_Authors 1317 ⟩
```

1333. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ recathrs.web 1313 ⟩
⟨ Define Records_Authors functions 1320 ⟩
```

1334. Contributors (cntrbtrs.web).

Log

[LDF 2006.09.26.] Created this file. It contains code formerly in `cntrbtrs.h` and `cntrbtrs.cpp`.

```
⟨ cntrbtrs.web 1334 ⟩ ≡
  static char id_string[] = "$Id:␣cntrbtrs.web,v␣1.6␣2006/10/23␣14:21:40␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1354.

1335. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1336. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1337. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1338. Contributors class declaration.

Log

[LDF 2006.07.26.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Contributors 1338 ⟩ ≡
  class Contributors : public CRecordset {
  public: long m_contributor_id;
    CStringA m_given_name;
    CStringA m_surname;
    CStringA m_prefix;
    LONGLONGm_id_number_ppn;
    ⟨ Declare Contributors functions 1340 ⟩
  };
```

This code is used in section 1353.

1339. Functions.

1340. Constructor.

```
⟨ Declare Contributors functions 1340 ⟩ ≡
  Contributors(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Contributors)
```

See also sections 1342, 1344, 1346, 1348, and 1350.
This code is used in section 1338.

1341.

```
⟨ Define Contributors functions 1341 ⟩ ≡
  IMPLEMENT_DYNAMIC(Contributors, CRecordset)
  Contributors :: Contributors(CDatabase *pdb)
  : CRecordset(pdb) {
    m_contributor_id = 0;
    m_given_name = "";
    m_surname = "";
    m_prefix = "";
    m_id_number_ppn = 0;
    m_nFields = 5;
    m_nDefaultType = dynaset;
  }
```

See also sections 1343, 1345, 1347, 1349, and 1351.
This code is used in section 1354.

1342. Get default connect. [LDF 2006.07.26.]

```
⟨ Declare Contributors functions 1340 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1343.

```
⟨ Define Contributors functions 1341 ⟩ +≡
  CString Contributors :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1344. Standard-SQL for Recordset. [LDF 2006.07.26.]

```
⟨ Declare Contributors functions 1340 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1345.

```
⟨ Define Contributors functions 1341 ⟩ +≡
  CString Contributors :: GetDefaultSQL( )
  {
    return _T("[dbo].[Contributors]");
  }
```

1346. Do Field Exchange. RFX-Unterstützung. [LDF 2006.07.26.]

```
⟨ Declare Contributors functions 1340 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1347.

```
⟨ Define Contributors functions 1341 ⟩ +≡
  void Contributors::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[contributor_id]"), m_contributor_id);
    RFX_Text(pFX, _T("[given_name]"), m_given_name);
    RFX_Text(pFX, _T("[surname]"), m_surname);
    RFX_Text(pFX, _T("[prefix]"), m_prefix);
    RFX_BigInt(pFX, _T("[id_number_ppn]"), m_id_number_ppn);
  }
```

1348. Assert Valid. [LDF 2006.07.26.]

```
⟨ Declare Contributors functions 1340 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid() const;
#endif
```

1349.

```
⟨ Define Contributors functions 1341 ⟩ +≡
#ifdef _DEBUG
  void Contributors::AssertValid() const
  {
    CRecordset::AssertValid();
  }
#endif
```

1350. Dump. [LDF 2006.07.26.]

```
⟨ Declare Contributors functions 1340 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1351.

```
⟨ Define Contributors functions 1341 ⟩ +≡
#ifdef _DEBUG
  void Contributors::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1352. Putting Contributors together.

1353. This is what gets written to `cntrbtrs.h`.

```
⟨ cntrbtrs.h 1353 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Contributors 1338 ⟩
```

1354. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ cntrbtrs.web 1334 ⟩
⟨ Define Contributors functions 1341 ⟩
```

1355. Records_Contributors (rccntrbt.web).

Log

[LDF 2006.09.26.] Created this file. It contains code formerly in `Records_Contributors.h` and `Records_Contributors.cp`

```
⟨ rccntrbt.web 1355 ⟩ ≡
  static char id_string[] = "$Id:␣rccntrbt.web,v␣1.5␣2006/10/23␣14:24:09␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1375.

1356. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1357. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1358. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1359. Records_Contributors class declaration.

Log

[LDF 2006.07.26.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records_Contributors 1359 ⟩ ≡
  class Records_Contributors : public CRecordset {
  public: long m_record_id;
    long m_contributor_id;
    ⟨ Declare Records_Contributors functions 1361 ⟩
  };
```

This code is used in section 1374.

1360. Functions.

1361. Constructor.

```
⟨ Declare Records_Contributors functions 1361 ⟩ ≡
  Records_Contributors(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records_Contributors)
```

See also sections 1363, 1365, 1367, 1369, and 1371.

This code is used in section 1359.

1362.

⟨ Define **Records_Contributors** functions 1362 ⟩ ≡
```
  IMPLEMENT_DYNAMIC(Records_Contributors, CRecordset)
  Records_Contributors::Records_Contributors(CDatabase *pdb)
  : CRecordset(pdb) {
    m_record_id = 0;
    m_contributor_id = 0;
    m_nFields = 2;
    m_nDefaultType = dynaset;
  }
```
See also sections 1364, 1366, 1368, 1370, and 1372.

This code is used in section 1375.

1363. Get default connect. [LDF 2006.07.26.]

⟨ Declare **Records_Contributors** functions 1361 ⟩ +≡
```
  virtual CString GetDefaultConnect();
```

1364.

⟨ Define **Records_Contributors** functions 1362 ⟩ +≡
```
  CString Records_Contributors::GetDefaultConnect()
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1365. Standard-SQL for Recordset. [LDF 2006.07.26.]

⟨ Declare **Records_Contributors** functions 1361 ⟩ +≡
```
  virtual CString GetDefaultSQL();
```

1366.

⟨ Define **Records_Contributors** functions 1362 ⟩ +≡
```
  CString Records_Contributors::GetDefaultSQL()
  {
    return _T("[dbo].[Records_Contributors]");
  }
```

1367. Do Field Exchange. RFX-Unterstützung. [LDF 2006.07.26.]

⟨ Declare **Records_Contributors** functions 1361 ⟩ +≡
```
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1368.

```
⟨ Define Records_Contributors functions 1362 ⟩ +≡
  void Records_Contributors::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[contributor_id]"), m_contributor_id);
  }
```

1369. Assert Valid. [LDF 2006.07.26.]

```
⟨ Declare Records_Contributors functions 1361 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1370.

```
⟨ Define Records_Contributors functions 1362 ⟩ +≡
#ifdef _DEBUG
  void Records_Contributors::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1371. Dump. [LDF 2006.07.26.]

```
⟨ Declare Records_Contributors functions 1361 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1372.

```
⟨ Define Records_Contributors functions 1362 ⟩ +≡
#ifdef _DEBUG
  void Records_Contributors::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1373. Putting Records_Contributors together.

1374. This is what gets written to `rccntrbt.h`.

```
⟨ rccntrbt.h 1374 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records_Contributors 1359 ⟩
```

1375. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ rccntrbt.web 1355 ⟩
⟨ Define Records_Contributors functions 1362 ⟩
```

1376. Main_Titles (mnttls.web).

Log

[LDF 2006.09.26.] Created this file. It contains code formerly in `mnttls.h` and `mnttls.cpp`.

```
⟨ mnttls.web 1376 ⟩ ≡
  static char id_string[] = "$Id:␣mnttls.web,v␣1.5␣2006/10/23␣14:22:34␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1396.

1377. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1378. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1379. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1380. Main_Titles class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

[LDF 2006.09.04.] Added the data member **long** *m_continuation*.

[LDF 2006.09.05.] Changed **long** *m_continuation* to *m_continuation_main_canonical_title*. Added **long** *m_continuation_ad*

```
⟨ Declare class Main_Titles 1380 ⟩ ≡
  class Main_Titles : public CRecordset {
  public: ⟨ Declare Main_Titles functions 1382 ⟩
    long m_main_title_id;
    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;
  };
```
This code is used in section 1395.

1381. Functions.

1382. Constructor.

Log

[LDF 2006.09.04.] Added the initialization of **long** *m_continuation*.

[LDF 2006.09.05.] Changed **long** *m_continuation* to *m_continuation_main_canonical_title*. Added **long** *m_continuation_ad*

```
⟨ Declare Main_Titles functions 1382 ⟩ ≡
  Main_Titles(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Main_Titles)
```
See also sections 1384, 1386, 1388, 1390, and 1392.

This code is used in section 1380.

1383.

```
⟨ Define Main_Titles functions 1383 ⟩ ≡
  IMPLEMENT_DYNAMIC(Main_Titles, CRecordset)
  Main_Titles::Main_Titles(CDatabase *pdb)
  : CRecordset(pdb) {
    m_main_title_id = 0;
    m_standard_text = "";
    m_main_canonical_title = "";
    m_continuation_main_canonical_title = 0;
    m_additions_main = "";
    m_continuation_additions_main = 0;
    m_additional_creator_main = "";
    m_parallel_canonical_title = "";
    m_additions_parallel = "";
    m_additional_creator_parallel = "";
    m_authorship = "";
    m_nFields = 11;
    m_nDefaultType = dynaset;
  }
```
See also sections 1385, 1387, 1389, 1391, and 1393.

This code is used in section 1396.

1384. Get default connect. [LDF 2006.08.24.]

```
⟨ Declare Main_Titles functions 1382 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1385.

```
⟨ Define Main_Titles functions 1383 ⟩ +≡
  CString Main_Titles::GetDefaultConnect()
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1386. Standard-SQL for Recordset. [LDF 2006.08.24.]

```
⟨ Declare Main_Titles functions 1382 ⟩ +≡
  virtual CString GetDefaultSQL();
```

1387.

```
⟨ Define Main_Titles functions 1383 ⟩ +≡
  CString Main_Titles::GetDefaultSQL()
  {
    return _T("[dbo].[Main_Titles]");
  }
```

1388. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

```
⟨ Declare Main_Titles functions 1382 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1389.

```
⟨ Define Main_Titles functions 1383 ⟩ +≡
  void Main_Titles::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[main_title_id]"), m_main_title_id);
    RFX_Text(pFX, _T("[standard_text]"), m_standard_text, 512);
    RFX_Text(pFX, _T("[main_canonical_title]"), m_main_canonical_title, 512);
    RFX_Long(pFX, _T("[continuation_main_canonical_title]"),
        m_continuation_main_canonical_title);
    RFX_Text(pFX, _T("[additions_main]"), m_additions_main, 512);
    RFX_Long(pFX, _T("[continuation_additions_main]"), m_continuation_additions_main);
    RFX_Text(pFX, _T("[additional_creator_main]"), m_additional_creator_main, 512);
    RFX_Text(pFX, _T("[parallel_canonical_title]"), m_parallel_canonical_title, 512);
    RFX_Text(pFX, _T("[additions_parallel]"), m_additions_parallel, 512);
    RFX_Text(pFX, _T("[additional_creator_parallel]"), m_additional_creator_parallel, 512);
    RFX_Text(pFX, _T("[authorship]"), m_authorship, 512);
  }
```

1390. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Main_Titles functions 1382 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid() const;
#endif
```

1391.

```
⟨ Define Main_Titles functions 1383 ⟩ +≡
#ifdef _DEBUG
  void Main_Titles::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1392. Dump. [LDF 2006.08.24.]

```
⟨ Declare Main_Titles functions 1382 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1393.

```
⟨ Define Main_Titles functions 1383 ⟩ +≡
#ifdef _DEBUG
  void Main_Titles::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1394. Putting Main_Titles together.

1395. This is what gets written to `mnttls.h`.

```
⟨ mnttls.h 1395 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Main_Titles 1380 ⟩
```

1396. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ mnttls.web 1376 ⟩
⟨ Define Main_Titles functions 1383 ⟩
```

1397. Records_Main_Titles (rcmnttls.web).

Log

[LDF 2006.09.26.] Created this file. It contains code formerly in `Records_Main_Titles.h` and `Records_Main_Titles.cpp`.

```
⟨ rcmnttls.web 1397 ⟩ ≡
  static char id_string[] = "$Id:␣rcmnttls.web,v␣1.5␣2006/10/23␣14:24:24␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1417.

1398. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1399. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1400. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1401. Records_Main_Titles class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records_Main_Titles 1401 ⟩ ≡
  class Records_Main_Titles : public CRecordset {
  public: long m_record_id;
    long m_main_title_id;
    ⟨ Declare Records_Main_Titles functions 1403 ⟩
  };
```

This code is used in section 1416.

1402. Functions.

1403. Constructor.

```
⟨ Declare Records_Main_Titles functions 1403 ⟩ ≡
  Records_Main_Titles(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records_Main_Titles)
```

See also sections 1405, 1407, 1409, 1411, and 1413.

This code is used in section 1401.

1404.

⟨ Define **Records_Main_Titles** functions 1404 ⟩ ≡
```
IMPLEMENT_DYNAMIC(Records_Main_Titles, CRecordset)
Records_Main_Titles::Records_Main_Titles(CDatabase *pdb)
: CRecordset(pdb) {
  m_record_id = 0;
  m_main_title_id = 0;
  m_nFields = 2;
  m_nDefaultType = dynaset;
}
```
See also sections 1406, 1408, 1410, 1412, and 1414.

This code is used in section 1417.

1405. Get default connect. [LDF 2006.08.24.]

⟨ Declare **Records_Main_Titles** functions 1403 ⟩ +≡
```
virtual CString GetDefaultConnect();
```

1406.

⟨ Define **Records_Main_Titles** functions 1404 ⟩ +≡
```
CString Records_Main_Titles::GetDefaultConnect()
{
  return _T(DATABASE_CONNECTION_STRING);
}
```

1407. Standard-SQL for Recordset. [LDF 2006.08.24.]

⟨ Declare **Records_Main_Titles** functions 1403 ⟩ +≡
```
virtual CString GetDefaultSQL();
```

1408.

⟨ Define **Records_Main_Titles** functions 1404 ⟩ +≡
```
CString Records_Main_Titles::GetDefaultSQL()
{
  return _T("[dbo].[Records_Main_Titles]");
}
```

1409. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

⟨ Declare **Records_Main_Titles** functions 1403 ⟩ +≡
```
virtual void DoFieldExchange(CFieldExchange *pFX);
```

1410.

```
⟨ Define Records_Main_Titles functions 1404 ⟩ +≡
  void Records_Main_Titles :: DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange :: outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[main_title_id]"), m_main_title_id);
  }
```

1411. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Records_Main_Titles functions 1403 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1412.

```
⟨ Define Records_Main_Titles functions 1404 ⟩ +≡
#ifdef _DEBUG
  void Records_Main_Titles :: AssertValid( ) const
  {
    CRecordset :: AssertValid( );
  }
#endif
```

1413. Dump. [LDF 2006.08.24.]

```
⟨ Declare Records_Main_Titles functions 1403 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1414.

```
⟨ Define Records_Main_Titles functions 1404 ⟩ +≡
#ifdef _DEBUG
  void Records_Main_Titles :: Dump(CDumpContext &dc) const
  {
    CRecordset :: Dump(dc);
  }
#endif
```

1415. Putting Records_Main_Titles together.

1416. This is what gets written to rcmnttls.h.

```
⟨ rcmnttls.h 1416 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records_Main_Titles 1401 ⟩
```

1417. This is what gets compiled.

⟨ Include files 13 ⟩
⟨ `rcmnttls.web` 1397 ⟩
⟨ Define **Records_Main_Titles** functions 1404 ⟩

1418. Content_Summaries (`contsums.web`).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in `contsums.h` and `contsums.cpp`.

⟨ `contsums.web` 1418 ⟩ ≡
  static char *id_string*[] = "$Id:␣contsums.web,v␣1.5␣2006/10/23␣14:21:50␣Administrator␣Exp␣$";

This code is cited in sections 7 and 8.
This code is used in section 1438.

1419. Preprocessor macro calls.

⟨ Preprocessor macro calls 10 ⟩ +≡
#**pragma** *once*

1420. using declarations for namespaces.

⟨ **using** declarations for namespaces 191 ⟩ +≡
  using namespace std;

1421. Include files.

⟨ Include files 13 ⟩ +≡
#**include** "stdafx.h"

1422. Content_Summaries class declaration.

Log

[LDF 2006.09.07.] Added this declaration. It was generated by Visual Studio.

⟨ Declare **class Content_Summaries** 1422 ⟩ ≡
  class Content_Summaries : **public CRecordset** {
  public: **long** *m_content_summary_id*;
    long *m_record_id*;
    long *m_continuation*;
    CStringA *m_content_summary*;
    ⟨ Declare **Content_Summaries** functions 1424 ⟩
  };

This code is used in section 1437.

1423. Functions.

1424. Constructor.

⟨ Declare **Content_Summaries** functions 1424 ⟩ ≡
  Content_Summaries(**CDatabase** **pDatabase* = NULL);
  DECLARE_DYNAMIC(**Content_Summaries**)

See also sections 1426, 1428, 1430, 1432, and 1434.
This code is used in section 1422.

1425.

```
⟨ Define Content_Summaries functions 1425 ⟩ ≡
  IMPLEMENT_DYNAMIC(Content_Summaries, CRecordset)
  Content_Summaries::Content_Summaries(CDatabase *pdb)
  : CRecordset(pdb) {
    m_content_summary_id = 0;
    m_record_id = 0;
    m_continuation = 0;
    m_content_summary = "";
    m_nFields = 4;
    m_nDefaultType = dynaset;
  }
```

See also sections 1427, 1429, 1431, 1433, and 1435.

This code is used in section 1438.

1426. Get default connect. [LDF 2006.09.07.]

```
⟨ Declare Content_Summaries functions 1424 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1427.

```
⟨ Define Content_Summaries functions 1425 ⟩ +≡
  CString Content_Summaries::GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1428. Standard-SQL for Recordset. [LDF 2006.09.07.]

```
⟨ Declare Content_Summaries functions 1424 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1429.

```
⟨ Define Content_Summaries functions 1425 ⟩ +≡
  CString Content_Summaries::GetDefaultSQL( )
  {
    return _T("[dbo].[Content_Summaries]");
  }
```

1430. Do Field Exchange. RFX-Unterstützung. [LDF 2006.09.07.]

```
⟨ Declare Content_Summaries functions 1424 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1431.

```
⟨ Define Content_Summaries functions 1425 ⟩ +≡
  void Content_Summaries::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[content_summary_id]"), m_content_summary_id);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[continuation]"), m_continuation);
    RFX_Text(pFX, _T("[content_summary]"), m_content_summary, 1024);
  }
```

1432. Assert Valid. [LDF 2006.09.07.]

```
⟨ Declare Content_Summaries functions 1424 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1433.

```
⟨ Define Content_Summaries functions 1425 ⟩ +≡
#ifdef _DEBUG
  void Content_Summaries::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1434. Dump. [LDF 2006.09.07.]

```
⟨ Declare Content_Summaries functions 1424 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1435.

```
⟨ Define Content_Summaries functions 1425 ⟩ +≡
#ifdef _DEBUG
  void Content_Summaries::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1436. Putting Content_Summaries together.

1437. This is what gets written to `contsums.h`.

```
⟨ contsums.h 1437 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Content_Summaries 1422 ⟩
```

1438. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ contsums.web 1418 ⟩
⟨ Define Content_Summaries functions 1425 ⟩
```

1439. Languages (language.web).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in `language.h` and `language.cpp`.

```
⟨ language.web 1439 ⟩ ≡
  static char id_string[] = "$Id:␣language.web,v␣1.5␣2006/10/23␣14:22:25␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1459.

1440. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1441. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1442. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1443. Languages class declaration.

Log

[LDF 2006.08.25.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Languages 1443 ⟩ ≡
  class Languages : public CRecordset {
  public: long m_language_id;
    CStringA m_language_name_english;
    CStringA m_language_name_german;
    CStringA m_language_abbrev;
    ⟨ Declare Languages functions 1445 ⟩
  };
```

This code is used in section 1458.

1444. Functions.

1445. Constructor.

```
⟨ Declare Languages functions 1445 ⟩ ≡
  Languages(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Languages)
```

See also sections 1447, 1449, 1451, 1453, and 1455.

This code is used in section 1443.

1446.

⟨ Define **Languages** functions 1446 ⟩ ≡
```
  IMPLEMENT_DYNAMIC(Languages, CRecordset)
  Languages :: Languages(CDatabase *pdb)
  : CRecordset(pdb) {
    m_language_id = 0;
    m_language_name_english = "";
    m_language_name_german = "";
    m_language_abbrev = "";
    m_nFields = 4;
    m_nDefaultType = dynaset;
  }
```
See also sections 1448, 1450, 1452, 1454, and 1456.

This code is used in section 1459.

1447. Get default connect. [LDF 2006.08.25.]

⟨ Declare **Languages** functions 1445 ⟩ +≡
```
  virtual CString GetDefaultConnect( );
```

1448.

⟨ Define **Languages** functions 1446 ⟩ +≡
```
  CString Languages :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1449. Standard-SQL for Recordset. [LDF 2006.08.25.]

⟨ Declare **Languages** functions 1445 ⟩ +≡
```
  virtual CString GetDefaultSQL( );
```

1450.

⟨ Define **Languages** functions 1446 ⟩ +≡
```
  CString Languages :: GetDefaultSQL( )
  {
    return _T("[dbo].[Languages]");
  }
```

1451. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.25.]

⟨ Declare **Languages** functions 1445 ⟩ +≡
```
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1452.

```
⟨ Define Languages functions 1446 ⟩ +≡
  void Languages::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX->SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[language_id]"), m_language_id);
    RFX_Text(pFX, _T("[language_name_english]"), m_language_name_english);
    RFX_Text(pFX, _T("[language_name_german]"), m_language_name_german);
    RFX_Text(pFX, _T("[language_abbrev]"), m_language_abbrev);
  }
```

1453. Assert Valid. [LDF 2006.08.25.]

```
⟨ Declare Languages functions 1445 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid() const;
#endif
```

1454.

```
⟨ Define Languages functions 1446 ⟩ +≡
#ifdef _DEBUG
  void Languages::AssertValid() const
  {
    CRecordset::AssertValid();
  }
#endif
```

1455. Dump. [LDF 2006.08.25.]

```
⟨ Declare Languages functions 1445 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1456.

```
⟨ Define Languages functions 1446 ⟩ +≡
#ifdef _DEBUG
  void Languages::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1457. Putting Languages together.

1458. This is what gets written to `language.h`.

```
⟨ language.h 1458 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Languages 1443 ⟩
```

1459. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ language.web 1439 ⟩
⟨ Define Languages functions 1446 ⟩
```

1460. Records_Languages (`reclang.web`).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in `reclang.h` and `reclang.cpp`.

[LDF 2006.10.10.] !! BUG FIX: Replaced the code for the data members. I'd forgotten that **Records_Languages** has more of them than the normal association table.

```
⟨ reclang.web 1460 ⟩ ≡
  static char id_string[] = "$Id:␣reclang.web,v␣1.5␣2006/10/23␣14:24:56␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1480.

1461. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1462. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1463. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1464. Records_Languages class declaration.

Log

[LDF 2006.08.25.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records_Languages 1464 ⟩ ≡
  class Records_Languages : public CRecordset {
  public: long m_record_id;
    long m_language_id;
    CStringA m_association_type;
    CStringA m_association_type_name;
    ⟨ Declare Records_Languages functions 1466 ⟩
  };
```

This code is used in section 1479.

1465. Functions.

1466. Constructor.

```
⟨ Declare Records_Languages functions 1466 ⟩ ≡
```

```
  Records_Languages(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records_Languages)
```

See also sections 1468, 1470, 1472, 1474, and 1476.

This code is used in section 1464.

1467.

```
⟨ Define Records_Languages functions 1467 ⟩ ≡
  IMPLEMENT_DYNAMIC(Records_Languages, CRecordset)
  Records_Languages::Records_Languages(CDatabase *pdb)
  : CRecordset(pdb) {
    m_record_id = 0;
    m_language_id = 0;
    m_association_type = "";
    m_association_type_name = "";
    m_nFields = 4;
    m_nDefaultType = dynaset;
  }
```

See also sections 1469, 1471, 1473, 1475, and 1477.

This code is used in section 1480.

1468. Get default connect. [LDF 2006.08.25.]

```
⟨ Declare Records_Languages functions 1466 ⟩ +≡
  virtual CString GetDefaultConnect();
```

1469.

```
⟨ Define Records_Languages functions 1467 ⟩ +≡
  CString Records_Languages::GetDefaultConnect()
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1470. Standard-SQL for Recordset. [LDF 2006.08.25.]

```
⟨ Declare Records_Languages functions 1466 ⟩ +≡
  virtual CString GetDefaultSQL();
```

1471.

```
⟨ Define Records_Languages functions 1467 ⟩ +≡
  CString Records_Languages::GetDefaultSQL()
  {
    return _T("[dbo].[Records_Languages]");
  }
```

1472. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.25.]

```
⟨ Declare Records_Languages functions 1466 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1473.

```
⟨ Define Records_Languages functions 1467 ⟩ +≡
  void Records_Languages::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[language_id]"), m_language_id);
    RFX_Text(pFX, _T("[association_type]"), m_association_type);
    RFX_Text(pFX, _T("[association_type_name]"), m_association_type_name);
  }
```

1474. Assert Valid. [LDF 2006.08.25.]

```
⟨ Declare Records_Languages functions 1466 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1475.

```
⟨ Define Records_Languages functions 1467 ⟩ +≡
#ifdef _DEBUG
  void Records_Languages::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1476. Dump. [LDF 2006.08.25.]

```
⟨ Declare Records_Languages functions 1466 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1477.

```
⟨ Define Records_Languages functions 1467 ⟩ +≡
#ifdef _DEBUG
  void Records_Languages::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1478. Putting Records_Languages together.

1479. This is what gets written to `reclang.h`.

```
⟨ reclang.h 1479 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records_Languages 1464 ⟩
```

1480. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ reclang.web 1460 ⟩
⟨ Define Records_Languages functions 1467 ⟩
```

1481. Subject_Types (subjtyps.web).

Log

[LDF 2006.10.05.] Created this file. It contains code formerly in subjtyps.h and subjtyps.cpp.

```
⟨ subjtyps.web 1481 ⟩ ≡
  static char id_string[] = "$Id:␣subjtyps.web,v␣1.4␣2006/10/23␣14:26:04␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1501.

1482. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1483. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1484. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1485. Subject_Types class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Subject_Types 1485 ⟩ ≡
  class Subject_Types : public CRecordset {
  public: long m_subject_type_id;
    CStringA m_indicator;
    CStringA m_description_german;
    CStringA m_description_english;
    BOOL m_pica3_800;
    BOOL m_pica3_51xx;
    ⟨ Declare Subject_Types functions 1487 ⟩
  };
```

This code is used in section 1500.

1486. Functions.

1487. Constructor.

```
⟨ Declare Subject_Types functions 1487 ⟩ ≡
  Subject_Types(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Subject_Types)
```

See also sections 1489, 1491, 1493, 1495, and 1497.
This code is used in section 1485.

1488.

```
⟨ Define Subject_Types functions 1488 ⟩ ≡
  IMPLEMENT_DYNAMIC(Subject_Types, CRecordset)
  Subject_Types :: Subject_Types(CDatabase *pdb)
  : CRecordset(pdb) {
    m_subject_type_id = 0;
    m_indicator = "";
    m_description_german = "";
    m_description_english = "";
    m_pica3_800 = FALSE;
    m_pica3_51xx = FALSE;
    m_nFields = 6;
    m_nDefaultType = dynaset;
  }
```

See also sections 1490, 1492, 1494, 1496, and 1498.
This code is used in section 1501.

1489. Get default connect. [LDF 2006.08.24.]

```
⟨ Declare Subject_Types functions 1487 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1490.

```
⟨ Define Subject_Types functions 1488 ⟩ +≡
  CString Subject_Types :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1491. Standard-SQL for Recordset. [LDF 2006.08.24.]

```
⟨ Declare Subject_Types functions 1487 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1492.

```
⟨ Define Subject_Types functions 1488 ⟩ +≡
  CString Subject_Types :: GetDefaultSQL( )
  {
    return _T("[dbo].[Subject_Types]");
  }
```

1493. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

```
⟨ Declare Subject_Types functions 1487 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1494.

```
⟨ Define Subject_Types functions 1488 ⟩ +≡
  void Subject_Types::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[subject_type_id]"), m_subject_type_id);
    RFX_Text(pFX, _T("[indicator]"), m_indicator);
    RFX_Text(pFX, _T("[description_german]"), m_description_german);
    RFX_Text(pFX, _T("[description_english]"), m_description_english);
    RFX_Bool(pFX, _T("[pica3_800]"), m_pica3_800);
    RFX_Bool(pFX, _T("[pica3_51xx]"), m_pica3_51xx);
  }
```

1495. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Subject_Types functions 1487 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1496.

```
⟨ Define Subject_Types functions 1488 ⟩ +≡
#ifdef _DEBUG
  void Subject_Types::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1497. Dump. [LDF 2006.08.24.]

```
⟨ Declare Subject_Types functions 1487 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1498.

```
⟨ Define Subject_Types functions 1488 ⟩ +≡
#ifdef _DEBUG
  void Subject_Types::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1499. Putting Subject_Types together.

1500. This is what gets written to `subjtyps.h`.

```
⟨ subjtyps.h 1500 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Subject_Types 1485 ⟩
```

1501. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ subjtyps.web 1481 ⟩
⟨ Define Subject_Types functions 1488 ⟩
```

1502. Subjects (subjects.web).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in subjects.h and subjects.cpp.

```
⟨ subjects.web 1502 ⟩ ≡
  static char id_string[] = "$Id:␣subjects.web,v␣1.4␣2006/10/23␣14:25:57␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1522.

1503. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1504. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1505. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1506. Subjects class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Subjects 1506 ⟩ ≡
  class Subjects : public CRecordset {
  public: long m_subject_id;
    long m_subject_type_id;
    CStringA m_subject;
    LONGLONGm_id_number_ppn;
    int m_chain_number;
    int m_chain_link_number;
    CStringA m_chain_info;
    ⟨ Declare Subjects functions 1508 ⟩
  };
```

This code is used in section 1521.

1507. Functions.

1508. Constructor.

```
⟨ Declare Subjects functions 1508 ⟩ ≡
  Subjects(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Subjects)
```

See also sections 1510, 1512, 1514, 1516, and 1518.

This code is used in section 1506.

1509.

```
⟨ Define Subjects functions 1509 ⟩ ≡
  IMPLEMENT_DYNAMIC(Subjects, CRecordset)
  Subjects::Subjects(CDatabase *pdb)
  : CRecordset(pdb) {
    m_subject_id = 0;
    m_subject_type_id = 0;
    m_subject = "";
    m_id_number_ppn = 0;
    m_chain_number = 0;
    m_chain_link_number = 0;
    m_chain_info = "";
    m_nFields = 7;
    m_nDefaultType = dynaset;
  }
```

See also sections 1511, 1513, 1515, 1517, and 1519.

This code is used in section 1522.

1510. Get default connect. [LDF 2006.08.24.]

```
⟨ Declare Subjects functions 1508 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1511.

```
⟨ Define Subjects functions 1509 ⟩ +≡
  CString Subjects::GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1512. Standard-SQL for Recordset. [LDF 2006.08.24.]

```
⟨ Declare Subjects functions 1508 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1513.

```
⟨ Define Subjects functions 1509 ⟩ +≡
  CString Subjects::GetDefaultSQL( )
  {
    return _T("[dbo].[Subjects]");
  }
```

1514. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

```
⟨ Declare Subjects functions 1508 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1515.

```
⟨ Define Subjects functions 1509 ⟩ +≡
  void Subjects::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[subject_id]"), m_subject_id);
    RFX_Long(pFX, _T("[subject_type_id]"), m_subject_type_id);
    RFX_Text(pFX, _T("[subject]"), m_subject);
    RFX_BigInt(pFX, _T("[id_number_ppn]"), m_id_number_ppn);
    RFX_Int(pFX, _T("[chain_number]"), m_chain_number);
    RFX_Int(pFX, _T("[chain_link_number]"), m_chain_link_number);
    RFX_Text(pFX, _T("[chain_info]"), m_chain_info, 256);
  }
```

1516. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Subjects functions 1508 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1517.

```
⟨ Define Subjects functions 1509 ⟩ +≡
#ifdef _DEBUG
  void Subjects::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1518. Dump. [LDF 2006.08.24.]

```
⟨ Declare Subjects functions 1508 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1519.

```
⟨ Define Subjects functions 1509 ⟩ +≡
#ifdef _DEBUG
  void Subjects::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1520. Putting Subjects together.

1521. This is what gets written to `subjects.h`.

```
⟨ subjects.h 1521 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Subjects 1506 ⟩
```

1522. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ subjects.web 1502 ⟩
⟨ Define Subjects functions 1509 ⟩
```

1523. Records_Subjects (recsubjs.web).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in `recsubjs.h` and `recsubjs.cpp`.

```
⟨ recsubjs.web 1523 ⟩ ≡
  static char id_string[] = "$Id:␣recsubjs.web,v␣1.5␣2006/10/23␣14:25:34␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1543.

1524. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1525. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1526. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1527. Records_Subjects class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records_Subjects 1527 ⟩ ≡
  class Records_Subjects : public CRecordset {
  public: long m_record_id;
    long m_subject_id;
    ⟨ Declare Records_Subjects functions 1529 ⟩
  };
```

This code is used in section 1542.

1528. Functions.

1529. Constructor.

```
⟨ Declare Records_Subjects functions 1529 ⟩ ≡
  Records_Subjects(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records_Subjects)
```

See also sections 1531, 1533, 1535, 1537, and 1539.

This code is used in section 1527.

1530.

```
⟨ Define Records_Subjects functions 1530 ⟩ ≡
  IMPLEMENT_DYNAMIC(Records_Subjects, CRecordset)
  Records_Subjects::Records_Subjects(CDatabase *pdb)
  : CRecordset(pdb) {
    m_record_id = 0;
    m_subject_id = 0;
    m_nFields = 2;
    m_nDefaultType = dynaset;
  }
```

See also sections 1532, 1534, 1536, 1538, and 1540.

This code is used in section 1543.

1531. Get default connect. [LDF 2006.08.24.]

```
⟨ Declare Records_Subjects functions 1529 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1532.

```
⟨ Define Records_Subjects functions 1530 ⟩ +≡
  CString Records_Subjects::GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1533. Standard-SQL for Recordset. [LDF 2006.08.24.]

```
⟨ Declare Records_Subjects functions 1529 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1534.

```
⟨ Define Records_Subjects functions 1530 ⟩ +≡
  CString Records_Subjects::GetDefaultSQL( )
  {
    return _T("[dbo].[Records_Subjects]");
  }
```

1535. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

```
⟨ Declare Records_Subjects functions 1529 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1536.

```
⟨ Define Records_Subjects functions 1530 ⟩ +≡
  void Records_Subjects::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[subject_id]"), m_subject_id);
  }
```

1537. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Records_Subjects functions 1529 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1538.

```
⟨ Define Records_Subjects functions 1530 ⟩ +≡
#ifdef _DEBUG
  void Records_Subjects::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1539. Dump. [LDF 2006.08.24.]

```
⟨ Declare Records_Subjects functions 1529 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1540.

```
⟨ Define Records_Subjects functions 1530 ⟩ +≡
#ifdef _DEBUG
  void Records_Subjects::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1541. Putting Records_Subjects together.

1542. This is what gets written to `recsubjs.h`.

```
⟨ recsubjs.h 1542 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records_Subjects 1527 ⟩
```

1543. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ recsubjs.web 1523 ⟩
⟨ Define Records_Subjects functions 1530 ⟩
```

1544. Permutation_Patterns (prmpttrn.web).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in `prmpttrn.h` and `prmpttrn.cpp`.

```
⟨ prmpttrn.web 1544 ⟩ ≡
  static char id_string[] = "$Id:␣prmpttrn.web,v␣1.5␣2006/10/23␣14:23:38␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1564.

1545. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1546. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1547. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1548. Permutation_Patterns class declaration.

Log

[LDF 2006.08.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Permutation_Patterns 1548 ⟩ ≡
  class Permutation_Patterns : public CRecordset {
  public: long m_permutation_pattern_id;
    long m_record_id;
    long m_subject_id_start;
    long m_subject_id_end;
    long m_chain_number;
    CStringA m_permutation_pattern;
    ⟨ Declare Permutation_Patterns functions 1550 ⟩
  };
```

This code is used in section 1563.

1549. Functions.

1550. Constructor.

```
⟨ Declare Permutation_Patterns functions 1550 ⟩ ≡
  Permutation_Patterns(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Permutation_Patterns)
```

See also sections 1552, 1554, 1556, 1558, and 1560.
This code is used in section 1548.

1551.

```
⟨ Define Permutation_Patterns functions 1551 ⟩ ≡
  IMPLEMENT_DYNAMIC(Permutation_Patterns, CRecordset)
  Permutation_Patterns :: Permutation_Patterns(CDatabase *pdb)
  : CRecordset(pdb) {
    m_permutation_pattern_id = 0;
    m_record_id = 0;
    m_subject_id_start = 0;
    m_subject_id_end = 0;
    m_chain_number = 0;
    m_permutation_pattern = "";
    m_nFields = 6;
    m_nDefaultType = dynaset;
  }
```

See also sections 1553, 1555, 1557, 1559, and 1561.
This code is used in section 1564.

1552. Get default connect. [LDF 2006.08.24.]

```
⟨ Declare Permutation_Patterns functions 1550 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1553.

```
⟨ Define Permutation_Patterns functions 1551 ⟩ +≡
  CString Permutation_Patterns :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1554. Standard-SQL for Recordset. [LDF 2006.08.24.]

```
⟨ Declare Permutation_Patterns functions 1550 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1555.

```
⟨ Define Permutation_Patterns functions 1551 ⟩ +≡
  CString Permutation_Patterns :: GetDefaultSQL( )
  {
    return _T("[dbo].[Permutation_Patterns]");
  }
```

1556. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.24.]

```
⟨ Declare Permutation_Patterns functions 1550 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1557.

```
⟨ Define Permutation_Patterns functions 1551 ⟩ +≡
  void Permutation_Patterns :: DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange :: outputColumn);
    RFX_Long(pFX, _T("[permutation_pattern_id]"), m_permutation_pattern_id);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[subject_id_start]"), m_subject_id_start);
    RFX_Long(pFX, _T("[subject_id_end]"), m_subject_id_end);
    RFX_Long(pFX, _T("[chain_number]"), m_chain_number);
    RFX_Text(pFX, _T("[permutation_pattern]"), m_permutation_pattern);
  }
```

1558. Assert Valid. [LDF 2006.08.24.]

```
⟨ Declare Permutation_Patterns functions 1550 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1559.

```
⟨ Define Permutation_Patterns functions 1551 ⟩ +≡
#ifdef _DEBUG
  void Permutation_Patterns :: AssertValid( ) const
  {
    CRecordset :: AssertValid( );
  }
#endif
```

1560. Dump. [LDF 2006.08.24.]

```
⟨ Declare Permutation_Patterns functions 1550 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1561.

```
⟨ Define Permutation_Patterns functions 1551 ⟩ +≡
#ifdef _DEBUG
  void Permutation_Patterns :: Dump(CDumpContext &dc) const
  {
    CRecordset :: Dump(dc);
  }
#endif
```

1562. Putting Permutation_Patterns together.

1563. This is what gets written to `prmpttrn.h`.

```
⟨ prmpttrn.h 1563 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Permutation_Patterns 1548 ⟩
```

1564. This is what gets compiled.

⟨ Include files 13 ⟩
⟨ `prmpttrn.web` 1544 ⟩
⟨ Define **Permutation_Patterns** functions 1551 ⟩

1565. Remote_Access (`rmaccess.web`).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in `rmaccess.h` and `rmaccess.cpp`.

⟨ `rmaccess.web` 1565 ⟩ ≡
static char *id_string*[] = `"$Id:␣rmaccess.web,v␣1.4␣2006/10/23␣14:25:42␣Administrator␣Exp␣$"`;

This code is cited in sections 7 and 8.

This code is used in section 1585.

1566. Preprocessor macro calls.

⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma *once*

1567. using declarations for namespaces.

⟨ **using** declarations for namespaces 191 ⟩ +≡
using namespace std;

1568. Include files.

⟨ Include files 13 ⟩ +≡
#include `"stdafx.h"`

1569. Remote_Access class declaration.

Log

[LDF 2006.08.25.] Added this declaration. It was generated by Visual Studio.

⟨ Declare **class Remote_Access** 1569 ⟩ ≡
```
class Remote_Access : public CRecordset {
public: 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;
  ⟨ Declare Remote_Access functions 1571 ⟩
};
```
This code is used in section 1584.

1570. Functions.

1571. Constructor.

⟨ Declare **Remote_Access** functions 1571 ⟩ ≡

```
  Remote_Access(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Remote_Access)
```

See also sections 1573, 1575, 1577, 1579, and 1581.
This code is used in section 1569.

1572.

```
⟨ Define Remote_Access functions 1572 ⟩ ≡
  IMPLEMENT_DYNAMIC(Remote_Access, CRecordset)
  Remote_Access::Remote_Access(CDatabase *pdb)
  : CRecordset(pdb) {
    m_remote_access_id = 0;
    m_license_indicator = FALSE;
    m_format_type = "";
    m_web_display_text = "";
    m_URL = "";
    m_URN = "";
    m_internal_remarks = "";
    m_nFields = 7;
    m_nDefaultType = dynaset;
  }
```

See also sections 1574, 1576, 1578, 1580, and 1582.
This code is used in section 1585.

1573. Get default connect. [LDF 2006.08.25.]

```
⟨ Declare Remote_Access functions 1571 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1574.

```
⟨ Define Remote_Access functions 1572 ⟩ +≡
  CString Remote_Access::GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1575. Standard-SQL for Recordset. [LDF 2006.08.25.]

```
⟨ Declare Remote_Access functions 1571 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1576.

```
⟨ Define Remote_Access functions 1572 ⟩ +≡
  CString Remote_Access::GetDefaultSQL( )
  {
    return _T("[dbo].[Remote_Access]");
  }
```

1577. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.25.]

```
⟨ Declare Remote_Access functions 1571 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1578.

```
⟨ Define Remote_Access functions 1572 ⟩ +≡
  void Remote_Access::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[remote_access_id]"), m_remote_access_id);
    RFX_Bool(pFX, _T("[license_indicator]"), m_license_indicator);
    RFX_Text(pFX, _T("[format_type]"), m_format_type);
    RFX_Text(pFX, _T("[web_display_text]"), m_web_display_text, 512);
    RFX_Text(pFX, _T("[URL]"), m_URL, 512);
    RFX_Text(pFX, _T("[URN]"), m_URN, 512);
    RFX_Text(pFX, _T("[internal_remarks]"), m_internal_remarks, 512);
  }
```

1579. Assert Valid. [LDF 2006.08.25.]

```
⟨ Declare Remote_Access functions 1571 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1580.

```
⟨ Define Remote_Access functions 1572 ⟩ +≡
#ifdef _DEBUG
  void Remote_Access::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1581. Dump. [LDF 2006.08.25.]

```
⟨ Declare Remote_Access functions 1571 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1582.

```
⟨ Define Remote_Access functions 1572 ⟩ +≡
#ifdef _DEBUG
  void Remote_Access::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1583. Putting Remote_Access together.

1584. This is what gets written to `rmaccess.h`.

```
⟨ rmaccess.h 1584 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Remote_Access 1569 ⟩
```

1585. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ rmaccess.web 1565 ⟩
⟨ Define Remote_Access functions 1572 ⟩
```

1586. Records_Remote_Access (`rcrmaccs.web`).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in `rcrmaccs.h` and `rcrmaccs.cpp`.

```
⟨ rcrmaccs.web 1586 ⟩ ≡
  static char id_string[] = "$Id:␣rcrmaccs.web,v␣1.5␣2006/10/23␣14:24:39␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1606.

1587. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1588. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1589. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1590. Records_Remote_Access class declaration.

Log

[LDF 2006.08.25.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records_Remote_Access 1590 ⟩ ≡
  class Records_Remote_Access : public CRecordset {
  public: long m_record_id;
    long m_remote_access_id;
    ⟨ Declare Records_Remote_Access functions 1592 ⟩
  };
```

This code is used in section 1605.

1591. Functions.

1592. Constructor.

```
⟨ Declare Records_Remote_Access functions 1592 ⟩ ≡
  Records_Remote_Access(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records_Remote_Access)
```

See also sections 1594, 1596, 1598, 1600, and 1602.

This code is used in section 1590.

1593.

```
⟨ Define Records_Remote_Access functions 1593 ⟩ ≡
  IMPLEMENT_DYNAMIC(Records_Remote_Access, CRecordset)
  Records_Remote_Access :: Records_Remote_Access(CDatabase *pdb)
  : CRecordset(pdb) {
    m_record_id = 0;
    m_remote_access_id = 0;
    m_nFields = 2;
    m_nDefaultType = dynaset;
  }
```

See also sections 1595, 1597, 1599, 1601, and 1603.

This code is used in section 1606.

1594. Get default connect. [LDF 2006.08.25.]

```
⟨ Declare Records_Remote_Access functions 1592 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1595.

```
⟨ Define Records_Remote_Access functions 1593 ⟩ +≡
  CString Records_Remote_Access :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1596. Standard-SQL for Recordset. [LDF 2006.08.25.]

```
⟨ Declare Records_Remote_Access functions 1592 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1597.

```
⟨ Define Records_Remote_Access functions 1593 ⟩ +≡
  CString Records_Remote_Access :: GetDefaultSQL( )
  {
    return _T("[dbo].[Records_Remote_Access]");
  }
```

1598. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.25.]

```
⟨ Declare Records_Remote_Access functions 1592 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1599.

```
⟨ Define Records_Remote_Access functions 1593 ⟩ +≡
  void Records_Remote_Access :: DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange :: outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[remote_access_id]"), m_remote_access_id);
  }
```

1600. Assert Valid. [LDF 2006.08.25.]

```
⟨ Declare Records_Remote_Access functions 1592 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1601.

```
⟨ Define Records_Remote_Access functions 1593 ⟩ +≡
#ifdef _DEBUG
  void Records_Remote_Access :: AssertValid( ) const
  {
    CRecordset :: AssertValid( );
  }
#endif
```

1602. Dump. [LDF 2006.08.25.]

```
⟨ Declare Records_Remote_Access functions 1592 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1603.

```
⟨ Define Records_Remote_Access functions 1593 ⟩ +≡
#ifdef _DEBUG
  void Records_Remote_Access :: Dump(CDumpContext &dc) const
  {
    CRecordset :: Dump(dc);
  }
#endif
```

1604. Putting Records_Remote_Access together.

1605. This is what gets written to `rcrmaccs.h`.

```
⟨ rcrmaccs.h 1605 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records_Remote_Access 1590 ⟩
```

1606. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ rcrmaccs.web 1586 ⟩
⟨ Define Records_Remote_Access functions 1593 ⟩
```

1607. Publishers (publshrs.web).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in publshrs.h and publshrs.cpp.

```
⟨ publshrs.web 1607 ⟩ ≡
  static char id_string[] = "$Id:␣publshrs.web,v␣1.4␣2006/10/23␣14:23:46␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1627.

1608. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1609. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1610. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1611. Publishers class declaration.

Log

[LDF 2006.08.28.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Publishers 1611 ⟩ ≡
  class Publishers : public CRecordset {
  public: long m_publisher_id;
    CStringA m_publisher_name;
    CStringA m_place;
    BOOL m_primary_info_source;
    ⟨ Declare Publishers functions 1613 ⟩
  };
```

This code is used in section 1626.

1612. Functions.

1613. Constructor.

```
⟨ Declare Publishers functions 1613 ⟩ ≡
  Publishers(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Publishers)
```

See also sections 1615, 1617, 1619, 1621, and 1623.

This code is used in section 1611.

1614.

```
⟨ Define Publishers functions 1614 ⟩ ≡
  IMPLEMENT_DYNAMIC(Publishers, CRecordset)
  Publishers::Publishers(CDatabase *pdb)
  : CRecordset(pdb) {
    m_publisher_id = 0;
    m_publisher_name = "";
    m_place = "";
    m_primary_info_source = FALSE;
    m_nFields = 4;
    m_nDefaultType = dynaset;
  }
```

See also sections 1616, 1618, 1620, 1622, and 1624.

This code is used in section 1627.

1615. Get default connect. [LDF 2006.08.28.]

```
⟨ Declare Publishers functions 1613 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1616.

```
⟨ Define Publishers functions 1614 ⟩ +≡
  CString Publishers::GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1617. Standard-SQL for Recordset. [LDF 2006.08.28.]

```
⟨ Declare Publishers functions 1613 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1618.

```
⟨ Define Publishers functions 1614 ⟩ +≡
  CString Publishers::GetDefaultSQL( )
  {
    return _T("[dbo].[Publishers]");
  }
```

1619. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.28.]

Log

[LDF 2006.09.11.] In the calls to *RFX_Text* for *m_publisher_name* and *m_publisher_place*: Added *nMaxLength* arguments with value 256.

```
⟨ Declare Publishers functions 1613 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1620.

```
⟨ Define Publishers functions 1614 ⟩ +≡
  void Publishers::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX->SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[publisher_id]"), m_publisher_id);
    RFX_Text(pFX, _T("[publisher_name]"), m_publisher_name, 256);
    RFX_Text(pFX, _T("[place]"), m_place, 256);
    RFX_Bool(pFX, _T("[primary_info_source]"), m_primary_info_source);
  }
```

1621. Assert Valid. [LDF 2006.08.28.]

```
⟨ Declare Publishers functions 1613 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1622.

```
⟨ Define Publishers functions 1614 ⟩ +≡
#ifdef _DEBUG
  void Publishers::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1623. Dump. [LDF 2006.08.28.]

```
⟨ Declare Publishers functions 1613 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1624.

```
⟨ Define Publishers functions 1614 ⟩ +≡
#ifdef _DEBUG
  void Publishers::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1625. Putting Publishers together.

1626. This is what gets written to `publshrs.h`.

```
⟨ publshrs.h 1626 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Publishers 1611 ⟩
```

1627. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ publshrs.web 1607 ⟩
⟨ Define Publishers functions 1614 ⟩
```

1628. Records_Publishers (recpubs.web).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in `recpubs.h` and `recpubs.cpp`.

```
⟨ recpubs.web 1628 ⟩ ≡
  static char id_string[] = "$Id:␣recpubs.web,v␣1.5␣2006/10/23␣14:25:25␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1648.

1629. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1630. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1631. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1632. Records_Publishers class declaration.

Log

[LDF 2006.08.28.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records_Publishers 1632 ⟩ ≡
  class Records_Publishers : public CRecordset {
  public: long m_record_id;
    long m_publisher_id;
    ⟨ Declare Records_Publishers functions 1634 ⟩
  };
```

This code is used in section 1647.

1633. Functions.

1634. Constructor.

```
⟨ Declare Records_Publishers functions 1634 ⟩ ≡
  Records_Publishers(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records_Publishers)
```

See also sections 1636, 1638, 1640, 1642, and 1644.

This code is used in section 1632.

1635.

```
⟨ Define Records_Publishers functions 1635 ⟩ ≡
  IMPLEMENT_DYNAMIC(Records_Publishers, CRecordset)
  Records_Publishers :: Records_Publishers(CDatabase *pdb)
  : CRecordset(pdb) {
    m_record_id = 0;
    m_publisher_id = 0;
    m_nFields = 2;
    m_nDefaultType = dynaset;
  }
```

See also sections 1637, 1639, 1641, 1643, and 1645.

This code is used in section 1648.

1636. Get default connect. [LDF 2006.08.28.]

```
⟨ Declare Records_Publishers functions 1634 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1637.

```
⟨ Define Records_Publishers functions 1635 ⟩ +≡
  CString Records_Publishers :: GetDefaultConnect( )
  { }
```

1638. Standard-SQL for Recordset. [LDF 2006.08.28.]

```
⟨ Declare Records_Publishers functions 1634 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1639.

```
⟨ Define Records_Publishers functions 1635 ⟩ +≡
  CString Records_Publishers :: GetDefaultSQL( )
  {
    return _T("[dbo].[Records_Publishers]");
  }
```

1640. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.28.]

```
⟨ Declare Records_Publishers functions 1634 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1641.

```
⟨ Define Records_Publishers functions 1635 ⟩ +≡
  void Records_Publishers::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[publisher_id]"), m_publisher_id);
  }
```

1642. Assert Valid. [LDF 2006.08.28.]

```
⟨ Declare Records_Publishers functions 1634 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1643.

```
⟨ Define Records_Publishers functions 1635 ⟩ +≡
#ifdef _DEBUG
  void Records_Publishers::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1644. Dump. [LDF 2006.08.28.]

```
⟨ Declare Records_Publishers functions 1634 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1645.

```
⟨ Define Records_Publishers functions 1635 ⟩ +≡
#ifdef _DEBUG
  void Records_Publishers::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1646. Putting Records_Publishers together.

1647. This is what gets written to `recpubs.h`.

```
⟨ recpubs.h 1647 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records_Publishers 1632 ⟩
```

1648. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ recpubs.web 1628 ⟩
⟨ Define Records_Publishers functions 1635 ⟩
```

1649. Database_Providers (dbprovs.web).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in dbprovs.h and dbprovs.cpp.

```
⟨ dbprovs.web 1649 ⟩ ≡
  static char id_string[] = "$Id:␣dbprovs.web,v␣1.5␣2006/10/23␣14:21:59␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1669.

1650. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1651. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1652. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1653. Database_Providers class declaration.

Log

[LDF 2006.08.28.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Database_Providers 1653 ⟩ ≡
  class Database_Providers : public CRecordset {
  public: long m_database_provider_id;
    CStringA m_database_provider_name;
    CStringA m_place;
    ⟨ Declare Database_Providers functions 1655 ⟩
  };
```

This code is used in section 1668.

1654. Functions.

1655. Constructor.

```
⟨ Declare Database_Providers functions 1655 ⟩ ≡
  Database_Providers(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Database_Providers)
```

See also sections 1657, 1659, 1661, 1663, and 1665.

This code is used in section 1653.

1656.

⟨ Define **Database_Providers** functions 1656 ⟩ ≡
```
IMPLEMENT_DYNAMIC(Database_Providers, CRecordset)
Database_Providers::Database_Providers(CDatabase *pdb)
: CRecordset(pdb) {
  m_database_provider_id = 0;
  m_database_provider_name = "";
  m_place = "";
  m_nFields = 3;
  m_nDefaultType = dynaset;
}
```
See also sections 1658, 1660, 1662, 1664, and 1666.
This code is used in section 1669.

1657. Get default connect. [LDF 2006.08.28.]

⟨ Declare **Database_Providers** functions 1655 ⟩ +≡
```
virtual CString GetDefaultConnect( );
```

1658.

⟨ Define **Database_Providers** functions 1656 ⟩ +≡
```
CString Database_Providers::GetDefaultConnect( )
{
  return _T(DATABASE_CONNECTION_STRING);
}
```

1659. Standard-SQL for Recordset. [LDF 2006.08.28.]

⟨ Declare **Database_Providers** functions 1655 ⟩ +≡
```
virtual CString GetDefaultSQL( );
```

1660.

⟨ Define **Database_Providers** functions 1656 ⟩ +≡
```
CString Database_Providers::GetDefaultSQL( )
{
  return _T("[dbo].[Database_Providers]");
}
```

1661. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.28.]

⟨ Declare **Database_Providers** functions 1655 ⟩ +≡
```
virtual void DoFieldExchange(CFieldExchange *pFX);
```

1662.

```
⟨ Define Database_Providers functions 1656 ⟩ +≡
  void Database_Providers::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[database_provider_id]"), m_database_provider_id);
    RFX_Text(pFX, _T("[database_provider_name]"), m_database_provider_name, 256);
    RFX_Text(pFX, _T("[place]"), m_place, 256);
  }
```

1663. Assert Valid. [LDF 2006.08.28.]

```
⟨ Declare Database_Providers functions 1655 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid( ) const;
#endif
```

1664.

```
⟨ Define Database_Providers functions 1656 ⟩ +≡
#ifdef _DEBUG
  void Database_Providers::AssertValid( ) const
  {
    CRecordset::AssertValid( );
  }
#endif
```

1665. Dump. [LDF 2006.08.28.]

```
⟨ Declare Database_Providers functions 1655 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1666.

```
⟨ Define Database_Providers functions 1656 ⟩ +≡
#ifdef _DEBUG
  void Database_Providers::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1667. Putting Database_Providers together.

1668. This is what gets written to `dbprovs.h`.

```
⟨ dbprovs.h 1668 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Database_Providers 1653 ⟩
```

1669. This is what gets compiled.

```
⟨ Include files 13 ⟩
⟨ dbprovs.web 1649 ⟩
⟨ Define Database_Providers functions 1656 ⟩
```

1670. Records_Database_Providers (rcdbprov.web).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in `rcdbprov.h` and `rcdbprov.cpp`.

```
⟨ rcdbprov.web 1670 ⟩ ≡
  static char id_string[] = "$Id:␣rcdbprov.web,v␣1.5␣2006/10/23␣14:24:17␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1690.

1671. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1672. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1673. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1674. Records_Database_Providers class declaration.

Log

[LDF 2006.08.28.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Records_Database_Providers 1674 ⟩ ≡
  class Records_Database_Providers : public CRecordset {
  public: long m_record_id;
    long m_database_provider_id;
    ⟨ Declare Records_Database_Providers functions 1676 ⟩
  };
```

This code is used in section 1689.

1675. Functions.

1676. Constructor.

```
⟨ Declare Records_Database_Providers functions 1676 ⟩ ≡
  Records_Database_Providers(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Records_Database_Providers)
```

See also sections 1678, 1680, 1682, 1684, and 1686.

This code is used in section 1674.

1677.

```
⟨ Define Records_Database_Providers functions 1677 ⟩ ≡
  IMPLEMENT_DYNAMIC(Records_Database_Providers, CRecordset)
  Records_Database_Providers :: Records_Database_Providers(CDatabase *pdb)
  : CRecordset(pdb) {
    m_record_id = 0;
    m_database_provider_id = 0;
    m_nFields = 2;
    m_nDefaultType = dynaset;
  }
```

See also sections 1679, 1681, 1683, 1685, and 1687.

This code is used in section 1690.

1678. Get default connect. [LDF 2006.08.28.]

```
⟨ Declare Records_Database_Providers functions 1676 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1679.

```
⟨ Define Records_Database_Providers functions 1677 ⟩ +≡
  CString Records_Database_Providers :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1680. Standard-SQL for Recordset. [LDF 2006.08.28.]

```
⟨ Declare Records_Database_Providers functions 1676 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1681.

```
⟨ Define Records_Database_Providers functions 1677 ⟩ +≡
  CString Records_Database_Providers :: GetDefaultSQL( )
  {
    return _T("[dbo].[Records_Database_Providers]");
  }
```

1682. Do Field Exchange. RFX-Unterstützung. [LDF 2006.08.28.]

```
⟨ Declare Records_Database_Providers functions 1676 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1683.

```
⟨ Define Records_Database_Providers functions 1677 ⟩ +≡
  void Records_Database_Providers::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX→SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[record_id]"), m_record_id);
    RFX_Long(pFX, _T("[database_provider_id]"), m_database_provider_id);
  }
```

1684. Assert Valid. [LDF 2006.08.28.]

```
⟨ Declare Records_Database_Providers functions 1676 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid() const;
#endif
```

1685.

```
⟨ Define Records_Database_Providers functions 1677 ⟩ +≡
#ifdef _DEBUG
  void Records_Database_Providers::AssertValid() const
  {
    CRecordset::AssertValid();
  }
#endif
```

1686. Dump. [LDF 2006.08.28.]

```
⟨ Declare Records_Database_Providers functions 1676 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1687.

```
⟨ Define Records_Database_Providers functions 1677 ⟩ +≡
#ifdef _DEBUG
  void Records_Database_Providers::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1688. Putting Records_Database_Providers together.

1689. This is what gets written to `rcdbprov.h`.

```
⟨ rcdbprov.h 1689 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Records_Database_Providers 1674 ⟩
```

1690. This is what gets compiled.

```
  ⟨ Include files 13 ⟩
  ⟨ rcdbprov.web 1670 ⟩
  ⟨ Define Records_Database_Providers functions 1677 ⟩
```

1691. Temp_IDs (tempids.web).

Log

[LDF 2006.09.27.] Created this file. It contains code formerly in `Temp_IDs.h` and `Temp_IDs.cpp`.

```
⟨ tempids.web 1691 ⟩ ≡
  static char id_string[] = "$Id:␣tempids.web,v␣1.5␣2006/10/23␣14:26:11␣Administrator␣Exp␣$";
```

This code is cited in sections 7 and 8.

This code is used in section 1711.

1692. Preprocessor macro calls.

```
⟨ Preprocessor macro calls 10 ⟩ +≡
#pragma once
```

1693. using declarations for namespaces.

```
⟨ using declarations for namespaces 191 ⟩ +≡
  using namespace std;
```

1694. Include files.

```
⟨ Include files 13 ⟩ +≡
#include "stdafx.h"
```

1695. Temp_IDs class declaration.

Log

[LDF 2006.07.24.] Added this declaration. It was generated by Visual Studio.

```
⟨ Declare class Temp_IDs 1695 ⟩ ≡
  class Temp_IDs : public CRecordset {
  public: long m_temp_id;
    ⟨ Declare Temp_IDs functions 1697 ⟩
  };
```

This code is used in section 1710.

1696. Functions.

1697. Constructor.

```
⟨ Declare Temp_IDs functions 1697 ⟩ ≡
  Temp_IDs(CDatabase *pDatabase = NULL);
  DECLARE_DYNAMIC(Temp_IDs)
```

See also sections 1699, 1701, 1703, 1705, and 1707.

This code is used in section 1695.

1698.

```
⟨ Define Temp_IDs functions 1698 ⟩ ≡
  IMPLEMENT_DYNAMIC(Temp_IDs, CRecordset)
  Temp_IDs :: Temp_IDs(CDatabase *pdb)
  : CRecordset(pdb) {
    m_temp_id = 0;
    m_nFields = 1;
    m_nDefaultType = dynaset;
  }
```

See also sections 1700, 1702, 1704, 1706, and 1708.
This code is used in section 1711.

1699. Get default connect. [LDF 2006.07.24.]

```
⟨ Declare Temp_IDs functions 1697 ⟩ +≡
  virtual CString GetDefaultConnect( );
```

1700.

```
⟨ Define Temp_IDs functions 1698 ⟩ +≡
  CString Temp_IDs :: GetDefaultConnect( )
  {
    return _T(DATABASE_CONNECTION_STRING);
  }
```

1701. Standard-SQL for Recordset. [LDF 2006.07.24.]

```
⟨ Declare Temp_IDs functions 1697 ⟩ +≡
  virtual CString GetDefaultSQL( );
```

1702.

```
⟨ Define Temp_IDs functions 1698 ⟩ +≡
  CString Temp_IDs :: GetDefaultSQL( )
  {
    return _T("[dbo].[Temp_IDs]");
  }
```

1703. Do Field Exchange. RFX-Unterstützung. [LDF 2006.07.24.]

```
⟨ Declare Temp_IDs functions 1697 ⟩ +≡
  virtual void DoFieldExchange(CFieldExchange *pFX);
```

1704.

```
⟨ Define Temp_IDs functions 1698 ⟩ +≡
  void Temp_IDs::DoFieldExchange(CFieldExchange *pFX)
  {
    pFX->SetFieldType(CFieldExchange::outputColumn);
    RFX_Long(pFX, _T("[temp_id]"), m_temp_id);
  }
```

1705. Assert Valid. [LDF 2006.07.24.]

```
⟨ Declare Temp_IDs functions 1697 ⟩ +≡
#ifdef _DEBUG
  virtual void AssertValid() const;
#endif
```

1706.

```
⟨ Define Temp_IDs functions 1698 ⟩ +≡
#ifdef _DEBUG
  void Temp_IDs::AssertValid() const
  {
    CRecordset::AssertValid();
  }
#endif
```

1707. Dump. [LDF 2006.07.24.]

```
⟨ Declare Temp_IDs functions 1697 ⟩ +≡
#ifdef _DEBUG
  virtual void Dump(CDumpContext &dc) const;
#endif
```

1708.

```
⟨ Define Temp_IDs functions 1698 ⟩ +≡
#ifdef _DEBUG
  void Temp_IDs::Dump(CDumpContext &dc) const
  {
    CRecordset::Dump(dc);
  }
#endif
```

1709. Putting Temp_IDs together.

1710. This is what gets written to `tempids.h`.

```
⟨ tempids.h 1710 ⟩ ≡
  ⟨ Preprocessor macro calls 10 ⟩
  ⟨ using declarations for namespaces 191 ⟩
  ⟨ Declare class Temp_IDs 1695 ⟩
```

1711. This is what gets compiled.

⟨ Include files 13 ⟩
⟨ `tempids.web` 1691 ⟩
⟨ Define **Temp_IDs** functions 1698 ⟩

1712. PICA Codes. [LDF 2006.09.21.]
The **Category_Container** and **Subcategory_Container** functions are ordered according to the Pica+ codes, so it's not necessary to sort the entries in `picapc.tex`. However, if the entries in `picacp.tex` are to be ordered according to the Pica3 codes, an external tool must be used to sort them. It will also be necessary to run TEX twice in order to ensure that the entries are up-to-date. [LDF 2006.09.21.]

To Do

[LDF 2006.09.21.] Add code to `ctgcntnr.web` and `sbctgcnt.web` for writing entries to `picapc.tex` and `picacp.tex`. Format them as tables with headings.

1713. Ordered by Pica+ Code.

1714. Ordered by Pica3 Code.

1715. GNU General Public License. [LDF 2006.10.23.]

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.

GNU GENERAL PUBLIC LICENSE
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

⟨ GNU General Public License 1715 ⟩ ≡ /∗ This section contains no C++ code. ∗/

This code is cited in section 2.

This code is used in section 1717.

1716. GNU Free Documentation License. [LDF 2006.10.23.]

2. VERBATIM COPYING

3. COPYING IN QUANTITY

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

⟨ GNU Free Documentation License 1716 ⟩ ≡ /∗ This section contains no C++ code. ∗/

This code is cited in section 2.

This code is used in section 1717.

1717. Include sections without C++ code. These sections contain no code, or only commented-out code. However, they must be included somewhere, or CWEAVE will issue warnings. [LDF 2006.09.27.]

There is no `resource.cpp`, so ⟨ `resource.web` 22 ⟩ doesn't contain a static *id_string* variable, as most or all of the other CWEB files do. [LDF 2006.10.23.]

⟨ `resource.web` 22 ⟩
⟨ GNU General Public License 1715 ⟩
⟨ GNU Free Documentation License 1716 ⟩

1718. Index.

IWF Metadata Harvester II
ZTest: The Program
Version 1.0
by Laurence D. Finston

October 2006