

Z39.50 Client

API DLL Guide

: For Developers

[KERIS Z39.50 Client API DLL Guide]

◆

KERIS Z39.50 Client
API DLL

. (2005 10)

< >

zapi2005.dll :	9가
yaz2005.dll :	() yaz toolkit
iconv2005.dll :	()KSC5601 UTF-8 Conversion

	MultiByte	UTF-8
	Char	wchar_t
	Strtok	wcstok
	Strcmp	wcscmp
	Strcpy	wcscpy
	Atoi	_wtoi
	"";) char *pPort = "5008";	L"";) wchar_t *pPort = L"5008";

KZConnect()

KZInitRequest()

KZSearch()

KZPresent()

KZGetBrief()

KZGetRecordType()

KZAddRecord()

KZFinish()

KZGetError()

KZConnect()

<p>▪ Synopsis</p> <pre>int KZConnect (wchar_t *pHost, wchar_t *pPort);</pre>
<p>▪ Function</p> <p>Server() Session Open .</p>
<p>▪ Parameters</p> <p>wchar_t *pHost Host IP address Domain name .) "zserver.riss4u.net"</p> <p>wchar_t *pPort Host TCP port .) "5008"</p>
<p>▪ Returns</p> <p>session (0), -1 return .</p>
<p>▪ Notes</p>

▷ **Example.**

```

Int iSid;
wchar_t *pHost = L"zserver.riss4u.net";
wchar_t *pPort = L"5008";

iSid = KZConnect(pHost, pPort);
// return iSid( ID) , KZInitRequest(), KZSearch(),
// KZGetBrief() .

// ERROR
if(iSid < 0) {
    //
    wchar_t szBuff[256];
    KZGetError(szBuff);

    wchar_t *token;
    int iErrCode;

```

```
wchar_t szDesc[256];

token = wcstok(szBuff, L"\n");
if(!wcscmp(token, L"N")) // network error
{
    token = wcstok(NULL, L"\n");
    iErrCode = _wtoi(token); // error code
    token = wcstok(NULL, L"\n");
    wcscpy(szDesc, token); // error description
}
}
```

KZInitRequest()

■ Synopsis	
<pre>int KZInitRequest(int iSid, wchar_t *pRefId, wchar_t *pVer, wchar_t *pOpt, int iMsgSize, int iRecSize, wchar_t *pGroupid, wchar_t *pUserid, wchar_t *pPasswd, wchar_t *pImpId, wchar_t *pImpName, wchar_t *pImpVer);</pre>	
■ Function	
Server .	
■ Parameters	
int iSid	session .
KZConnect() return	.
wchar_t *pRefId	(operation) ID , Client Server / . , ID Search, Present ID . , (serial operation), NULL .) pRefId = L"";
wchar_t *pVer	Client Z39.50 Protocol Version . Version 2 "YY" , Version 3 "YYY" , Ver1 Ver2 .) pVer = L"YYY";
wchar_t *pOpt	Server .) "yynnnnnnynnnnnnnnn" -> Search, Present, ExtendedService(Update)
int iMsgSize	Byte . Server , size가 .) iMsgSize = 1048576;
int iRecSize	Present 가 , record size .) iRecSize = 1048576;
wchar_t *pGroupid	

<pre> pGroupid KERIS) "TESTID" wchar_t *pUserid , KERIS Z39.50) " " wchar_t *pPasswd) "TESTPWD" wchar_t *pImpId, wchar_t *pImpName, wchar_t *pImpVer Client) Unicat : "", "Unicat", "7.0" : "", "LAS ", " "</pre>															
<p>▪ Returns</p> <p>1, 0 return .</p>															
<p>▪ Notes</p> <table border="1"> <thead> <tr> <th><i>GroupID,</i></th> <th><i>Passwd</i></th> <th><i>UserID</i></th> </tr> </thead> <tbody> <tr> <td><i>"0/000000/TESTID/TESTPWD/</i></td> <td><i>"</i></td> <td><i>,</i></td> </tr> <tr> <td><i>pGroupid, pUserid, pPasswd</i></td> <td></td> <td></td> </tr> <tr> <td><i>pVer, pOpt</i></td> <td></td> <td><i>.(case insensitive)</i></td> </tr> <tr> <td><i>pGroupid, pUserid, pPasswd</i></td> <td></td> <td></td> </tr> </tbody> </table>	<i>GroupID,</i>	<i>Passwd</i>	<i>UserID</i>	<i>"0/000000/TESTID/TESTPWD/</i>	<i>"</i>	<i>,</i>	<i>pGroupid, pUserid, pPasswd</i>			<i>pVer, pOpt</i>		<i>.(case insensitive)</i>	<i>pGroupid, pUserid, pPasswd</i>		
<i>GroupID,</i>	<i>Passwd</i>	<i>UserID</i>													
<i>"0/000000/TESTID/TESTPWD/</i>	<i>"</i>	<i>,</i>													
<i>pGroupid, pUserid, pPasswd</i>															
<i>pVer, pOpt</i>		<i>.(case insensitive)</i>													
<i>pGroupid, pUserid, pPasswd</i>															

▷ Example.

```

int iSid = 0; // KZConnect() return .
wchar_t *pRefId = L"";
wchar_t *pVer = L"yyy";
wchar_t *pOpt = L"yynnnnnnnnnnnnnnnnnnnnn";
int iMsgSize = 1048576;
int iRecSize = 1048576;
wchar_t *pGroupid = L"TESTID";
wchar_t *pUserid = L" ";
wchar_t *pPasswd = L"TESTPWD";
wchar_t *pImpId = L"";
```

```
wchar_t *pImpName = L"UNICAT";
```

```
wchar_t *pImpVer = L"7.0";
```

```
iResult = KZInitRequest(pRefId, pVer, pOpt, iMsgSize, iRecSize,  
pGroupid, pUserid, pPasswd, pImpId, pImpName, pImpVer);
```

```
if (iResult > 1) {
```

```
    //
```

```
}
```

```
else {
```

```
    //
```

```
wchar_t szBuff[256];
```

```
KZGetError(szBuff);
```

```
wchar_t *token;
```

```
wchar_t szDesc[256];
```

```
token = wcstok(szBuff, L"\n");
```

```
if(!wcscmp(token, L"Z")) // Z3950 error
```

```
{
```

```
    token = wcstok(NULL, L"\n");
```

```
    wcscpy(szDesc, token); // error_code \t error_desc
```

```
    token = wcstok(NULL, L"\n");
```

```
    wcscpy(szDesc, token); // additional information
```

```
}
```

```
}
```

KZSearch()

■ Synopsis	
<pre>int KZSearch(int iSid, wchar_t *pDb, wchar_t *pQuery, wchar_t *pAttrSet, int *piCount, int sid);</pre>	
■ Function	
Z39.50 SearchRequest	Server
■ Parameters	
<p>int iSid SearchRequest session . KZConnect() return .</p> <p>wchar_t *pDb Database .) UALL, UBIB, UREF, UREQ, UAUT, USYS</p> <p>wchar_t *pQuery . (Example)) pQuery = L"and(or(or('BIB'[1,5001,2,3], 'REQ'[1,5001,2,3]), 'REF'[1,5001,2,3]), ' [1,4])";</p> <p>wchar_t *pAttrSet attribute set . (BIB-1)) pAttrSet = L"1.2.840.10003.3.1"</p> <p>int *piCount buffer</p>	
■ Returns	
1,	0 return .
■ Notes	
KERIS ,	"KERIS Z39.50 Configuration Guide"

▷ **Example.**

```
iSid = 0; // KZConnect() return .

// Query ,
// KERIS Z39.50 Configuration Guide .
// attributeSet ,
```

```
//
wchar_t *pDb = L"UALL";
wchar_t *pQuery = L"and(or(or('BIB'[1,5001,2,3], 'REQ'[1,5001,2,3]),
'REF'[1,5001,2,3]), '      '[1,4]))"; // Query
wchar_t *pAttrSet = L"1.2.840.10003.3.1";
// Attribute Sets : bib-1 => Z39.50 Protocol(OID)

iCount = 0; // KZSearch(          , iCount          .
iResult = KZSearch(iSid, pDb, pQuery, pAttrSet, &iCount);
```

▷ Query Example.

```
//          (          ) - DB      (UBIB, UREQ, UREF)
pQuery      =      L"and(or(or('BIB'[1,5001,2,3],          'REQ'[1,5001,2,3]),
'REF'[1,5001,2,3]), '      '[1,4]))";
// OCLC
pQuery = L"'999999'[1,5002,2,3]";
// Any      (          )
pQuery = L"and('BIB'[1,5001,2,3] '      '[1,1016,3,1]))";
//
pQuery      =      L"and(or(or('BIB'[1,5001,2,3],          'REQ'[1,5001,2,3]),
'REF'[1,5001,2,3]), '12345'[1,1007,3,1]))";
//
pQuery      =      L"and(and(and('BIB'[1,5001,2,3],          '2005'[1,31,2,2]),
'2005'[1,31,2,4]), '      '[1,4]))";
//
pQuery = L"and(and('BIB'[1,5001,2,3], 'kor'[1,54,2,3]), '      '[1,4]))";
//
pQuery      =      L"and(and(or(or('BIB'[1,5001,2,3],          'REQ'[1,5001,2,3]),
'REF'[1,5001,2,3]), 'ko'[1,59,2,3]), '      '[1,4]))";
//      +
pQuery      =      L"and(and(or(or('BIB'[1,5001,2,3],          'REQ'[1,5001,2,3]),
'REF'[1,5001,2,3]), '      '[1,4]), '      '[1,1003]))";
```

KZPresent()

■ Synopsis	
<pre>int KZPresent(int iSid, int iStart, int iCount, wchar_t *pRecSyntax, wchar_t *pElementSet);</pre>	
■ Function	
Z39.50 PresentRequest	Server
■ Parameters	
<p>int iSid PresentRequest session KZConnect() return</p> <p>int iStart Set present record 1 C/C++ 0 DLL iStart = 1; // s1 iStart = 11; // s2</p> <p>int iCount present record iCount = 10; // s1 : 1 10 10 Present iCount = 20; // s2 : 11 30 20 Present</p> <p>wchar_t *pRecSyntax record SUTRS pRecSyntax = L"1.2.840.10003.5.101"</p> <p>wchar_t *pElementSet Present Record KZGetBrief, "B" Brief Format KZGetRecordType() pElementSet = L"B";</p>	
■ Returns	
1, 0 return	
■ Notes	
Record Syntax	Z39.50 Protocol 'Z39.50 Object Identifiers' USMarc(Marc21) "1.2.840.10003.5.10"
SUTRS	'simple, unstructured text', KERIS 'Brief Format' 가

▷ Example.

```
iSid = 0; // KZConnect() return .
iStart = 1; // Start Rec-No
iCount = 10; // Count of Records
wchar_t *pRecSyntax = L"1.2.840.10003.5.101"; // SUTRS
wchar_t *pElementSet = L"B";
iResult = KZPresent(iSid, iStart, iCount, pRecSyntax, pElementSet);
```

KZGetBrief()

■ Synopsis	
<pre>int KZGetBrief (int iSid, int iRecNo, wchar_t *pBuff, int iBuff);</pre>	
■ Function	
Z39.50 PresentResponse 가 .	
■ Parameters	
int iSid	record 가 session . KZConnect() return .
int iRecNo	가 record . (set)
wchar_t *pBuff	buffer .
int iBuff	record size .
■ Returns	
1, 0 return .	
■ Notes	
(Brief Format) return . KZPresent() , record 가 record .	

▷ **Example.**

```
iSid = 0; // KZConnect() return .

// 10 Display , 2 .
// 가 300 , 11 20
10 records 가 .

// KZPresent()
iStart = 11; // Start Rec-No
iCount = 10; // Count of Records
pRecSyntax = L"1.2.840.10003.5.101";
```

```
pElementSet = L"B";  
iResult = KZPresent(iSid, iStart, iCount, pRecSyntax, pElementSet);  
if(iResult > 0)  
{  
    // OK : KZGetBrief() , Brief Format 가 .  
    wchar_t szBuff[32000];  
    int iBuff = 32000; // Buff Size  
  
    for(iRecNo = 11; iRecNo <= 20; ++iRecNo)  
    {  
        iResult = KZGetBrief (iSid, iRecNo, szBuff, iBuff);  
    }  
}
```

KZGetRecordType ()

■ Synopsis	
<pre>int KZGetRecordType(int iSid, int iRecNo, wchar_t *pBuff, int iBuff, wchar_t *pElementSet);</pre>	
■ Function	
Record 가 .	
■ Parameters	
int iSid	record 가 session .
	KZConnect() return .
int iRecNo	가 record . (set)
wchar_t *pBuff	buffer .
int iBuff	record size .
wchar_t *pElementSet	가 record type .
) "F" : MARC21("1.2.840.10003.5.10")
	* KZPresent .
■ Returns	
1, 0 return .	
■ Notes	
KZPresent() , pElementSet "F" Full	
Format(Marc Data) 가 .	
* : (BA), (BC), (BH), MARC(FH)	
"BA" :	SUTRS("1.2.840.10003.5.101")
"BC" :	SUTRS("1.2.840.10003.5.101")
"BH" :	SUTRS("1.2.840.10003.5.101")
"FH" :	MARC21("1.2.840.10003.5.10")

▷ Example.

```
iSid = 0; // KZConnect() return .
iRec = 1; // Rec-No: (ex: 1)
```

```
wchar_t szBuff[32000];  
int iBuff = 32000;          // Buff Size: (ex: 32000)  
wchar_t *ElementSet = L"F";
```

```
iResult = KZGetRecordType (iSid, iRec, szBuff, iBuff, pElementSet);
```

KZAddRecord()

■ Synopsis	
<code>int KZAddRecord (int iSid, wchar_t *pData);</code>	
■ Function	
Server	record 가 .
■ Parameters	
int iSid	session .
KZConnect() return	.
wchar_t *pData	가 record .
XML	,"KERIS Z39.50 Configuration Guide – Extended Service"
■ Returns	
1,	return .
■ Notes	
, 가, , , , , ,	
OCLC .	

▷ **Example.**

```

1.
iSid = 0; // KZConnect() return .
wchar_t *pData = L"<?xml
version="1.0" ?><db_name>UBIB</db_name><job_type>INS</job_type><entry_
date>20050321</entry_date><entry_time>154806</entry_time><entry_person
> </entry_person><entry_lib>999999</entry_lib><marc_data>00418nam
k200169 k
4500001001300000004001300013005000700026008004100033040002100074082001
8000952450037001132600036001503000020001867000012002068900011002189900
01900229- -000008516806-154806-970429s1977 ulk h000a
kor - a 211017c 211017- a 850.8192 20-00a 訥齋江 兩世貴稿 / d 朴江
著- a : b 韓國漢詩協會,c 1977- a 1冊 ; c 28 cm-1 a - h 245
- a 211017, - </marc_data></xml>";
iResult = KZAddRecord(iSid, pData);

```

2.

```
wchar_t *pData = L"<?xml
version="1.0" ?><db_name>UDWN</db_name><job_type>INS</job_type><bib_no
>45679</bib_no><entry_date>20050318</entry_date><entry_time>105934</en
try_time><entry_person>
</entry_person><entry_lib>999999</entry_lib><download_db_kind>UBIB</do
wnload_db_kind></xml>";
```

3. OCLC

```
wchar_t *pData = L"<?xml
version="1.0" ?><db_name>UOCL</db_name><job_type>INS</job_type><entry_
date>20050413</entry_date><entry_time>145732</entry_time><entry_person
>
</entry_person><entry_lib>999999</entry_lib><OCLC_title>
</OCLC_title><OCLC_isbn>1234123412</OCLC_isbn><OCLC_proc_status>P</OCL
C_proc_status></xml>";
```

▷

■

<XML >

```

<?xml version="1.0" ?>
<db_name> value </db_name> // DB
<job_type> value </job_type> //
<bib_no> value </bib_no> //
<mer_bib_no> // MER( ) ,
    <seq1> value </seq1>
    <seq2> value </seq2>
    <seq3> value </seq3> ...
</mer_bib_no>
<entry_date> value </entry_date> // yyyymmdd
<entry_time> value </entry_time> // hhmmss
<entry_person> value </entry_person> //
<entry_lib> value </entry_lib> // ( , 6 )
<remark> value </remark>
<local_system> value </local_system>
<marc_data> value </marc_data>
<abstract>
    <seq1> value </seq1>
    <seq2> value </seq2>
</abstract>
<toc> value </toc>
</xml>
    
```

* XML
Enter String

■

- db_name = UDWN
- job_type = INS

<XML > - 1

```

<?xml version="1.0" ?>
<db_name>UDWN</db_name>
<job_type>INS</job_type>
<bib_no>01768134</bib_no>
<entry_date>20020627</entry_date>
<entry_time>162521</entry_time>
<entry_person>      </entry_person>
<entry_lib>999999</entry_lib>
<local_system>UNICAT</local_system>
<download_db_kind>UBIB</download_db_kind>
</xml>
    
```

<XML > - 1

```

<?xml version="1.0" ?>
<db_name>UDWN</db_name>
<job_type>INS</job_type>
<bib_no>01768134</bib_no>
<entry_date>20020627</entry_date>
<entry_time>162521</entry_time>
<entry_person>      </entry_person>
<entry_lib>999999</entry_lib>
<local_system>UNICAT</local_system>
<download_db_kind>UREF</download_db_kind>
</xml>
    
```

■ OCLC

- db_name = UOCL
- job_type = INS, DEL

<XML > - OCLC

```

<?xml version="1.0" ?>
<db_name>UOCL</db_name>
<job_type>INS</job_type>
<entry_date>20020627</entry_date>
<entry_time>162521</entry_time>
<entry_person>      </entry_person>
<entry_lib>999999</entry_lib>
<local_system>UNICAT</local_system>
<OCLC_title> value </OCLC_title>
<OCLC_author> value </OCLC_author>
<OCLC_publisher> value </OCLC_publisher>
<OCLC_pub_year> value </OCLC_pub_year>
<OCLC_edition> value </OCLC_edition >
<OCLC_issn> value </OCLC_issn>
<OCLC_isbn> value </OCLC_isbn>
<OCLC_lccn> value </OCLC_lccn>
<OCLC_proc_status> P </OCLC_proc_status> //      (      P)
<OCLC_remark> value </OCLC_remark>
</xml>
    
```

<XML > - OCLC

```

<?xml version="1.0" ?>
<db_name>UPRI</db_name>
<job_type>DEL</job_type>
<entry_date>20020627</entry_date>
<entry_time>162521</entry_time>
<entry_person>      </entry_person>
<entry_lib>999999</entry_lib>
<local_system>UNICAT</local_system>
<OCLC_req_no>1234</OCLC_req_no> // OCLC
</xml>
    
```

KZFinish()

<p>▪ Synopsis</p> <pre>void KZFinish (int iSid);</pre>
<p>▪ Function</p> <p>Z-Session .</p>
<p>▪ Parameters</p> <p>int iSid</p> <p>session .</p> <p>KZConnect() return .</p>
<p>▪ Returns</p>
<p>▪ Notes</p> <p>KZConnect() , server .</p>

▷ **Example.**

```
iSid = 0; // KZConnect() return .
KZFinish(iSid);
```

KZGetError()

■ Synopsis	
void KZGetError(wchar_t *pBuff);	
■ Function	
가 , .	
■ Parameters	
wchar_t *pBuff	buffer .
■ Returns	
pBuff , .	
■ Notes	
<p>pBuff (\ n) 2 .</p> <p>1. : N(network error) or Z(Z3950 error)</p> <p>2. : N error code ,</p> <p>Z [error_code \ t error_description] 가 (\ t)</p> <p>3. : N error description ,</p> <p>Z , [additional information] , addinfo=" , "</p> <p>, 가 .</p> <p>N ,</p> <p>Z YAZ Toolkit Library ,</p> <p>"Bib-1 Diagnostics" .</p> <p>"Bib-1 Diagnostics" .</p>	

▷ **Example.**

```

iSid = KZConnect(pHost, pPort);
if(iSid < 0) {
    wchar_t szBuff[256];

    KZGetError (szBuff);

    wchar_t *token;
    
```

```
wchar_t szDesc[256];

token = wcstok(szBuff, L"\n");
if(!wcscmp(token, L"N")) // network error
{
    int iErrCode;
    token = wcstok(NULL, L"\n");
    iErrCode = _wtoi(token); // error code
    // ex) 10061
    token = wcstok(NULL, L"\n");
    wcscpy(szDesc, token); // error description
    // ex)
}
if(!wcscmp(token, L"Z")) // Z3950 error
{
    token = wcstok(NULL, L"\n");
    wcscpy(szDesc, token); // error_code \t error_desc
    // FAIL TO KZInitRequest()
    // ex) code=1014  Init/AC: Authentication System error
    // FAIL TO KZSearch()
    // ex) code=114  Unsupported Use attribute
    token = wcstok(NULL, L"\n");
    wcscpy(szDesc, token); // additional information
    // ex) addinfo=''
    // ex) addinfo='use : [2001]'
}
}
```

[]

Function Definition for C/C++

```
/*
*****
*****
*****
*/

//z39.50 API

typedef int (*FUNC_CONNECT)(wchar_t*, wchar_t*);
typedef int (*FUNC_INIT)(int, wchar_t *, wchar_t *, wchar_t *, int,
int, wchar_t *, wchar_t *, wchar_t *, wchar_t *, wchar_t *, wchar_t
*);
typedef int (*FUNC_SEARCH)(int, wchar_t *, wchar_t *, wchar_t *, int
*);
typedef int (*FUNC_PRESENT)(int, int, int, wchar_t *, wchar_t *);
typedef int (*FUNC_GET_BRIEF)(int, int, wchar_t *, int);
typedef int (*FUNC_GET_RECORD_TYPE)(int, int, wchar_t *, int, wchar_t
*);
typedef int (*FUNC_ADD_RECORD)(int, wchar_t *);
typedef void (*FUNC_CLOSE)(int);
typedef void (*FUNC_GET_ERROR)(wchar_t*);
```



```
    if(funcGetError != NULL)
    {
        wchar_t szBuff[256];
        funcGetError(szBuff);

        wchar_t *token;
        token = wcstok(szBuff, L"\n");
        if(token != NULL)
        {
            if(!wcscmp(token, L"N")) // network error
            {
                int iErrCode;
                token = wcstok(NULL, L"\n");
                iErrCode = _wtoi(token); // error code
                token = wcstok(NULL, L"\n");
                wscpy(szDesc, token); // error description
            }
        }
    }
}
```

```
// [ KZInitRequest ]
```

```
FUNC_INIT funcInit;
```

```
funcInit = (FUNC_INIT) GetProcAddress(m_hInst, "KZInitRequest");
```

```
if(funcInit == NULL)
```

```
    return;
```

```
wchar_t *pRefId = L"";
```

```
wchar_t *pVer = L"yyy";
```

```
wchar_t *pOpt = L" yynnnnnnnynnnnnnnnnn";
```

```
int iMsgSize = 1048576;
```

```
int iRecSize = 1048576;
```

```
wchar_t *pGroupid = L"U/KRIC";
```

```
wchar_t *pUserid = L"      ";
```

```
wchar_t *pPasswd = L"KRIC999";
```

```
wchar_t *pImpId = L"";
```

```
wchar_t *pImpName = L"UNICAT";
wchar_t *pImpVer = L"7.0";
iret = funcInit(m_nSessionID, pRefId, pVer, pOpt, iMsgSize, iRecSize,
               pGroupid, pUserid, pPasswd, pImpId, pImpName, pImpVer);
if(iret > 1)
    // OK - iret :          lib code
else
{
    // FAIL
```

```
// [ KZGetError ]
funcGetError = (FUNC_GET_ERROR) GetProcAddress(m_hInst,
"KZGetError");
if(funcGetError != NULL)
{
    wchar_t szBuff[256];
    funcGetError(szBuff);

    wchar_t *token;
    token = wcstok(szBuff, L"\n");
    if(token != NULL)
    {
        if(!wcscmp(token, L"Z") // Z3950 error
        {
            token = wcstok(NULL, L"\n");
            wcscpy(szDesc, token); // error_code \t
error_desc
            token = wcstok(NULL, L"\n");
            wcscpy(szDesc, token); // additional
information
        }
    }
}

// [ KZSearch ]
```

```
FUNC_SEARCH funcSearch;
funcSearch = (FUNC_SEARCH) GetProcAddress(m_hInst, "KZSearch");

wchar_t *pDb = L"UALL";
wchar_t *pQuery = L"and(or(or('BIB'[1,5001,2,3], 'REQ'[1,5001,2,3]),
'REF'[1,5001,2,3]), '[1,4])";
wchar_t *pAttrSet = L"1.2.840.10003.3.1";
int iCount = 0;
iret = funcSearch(m_nSessionID, pDb, pQuery, pAttrSet, &iCount);
if(iret == 1)
    // OK - iCount; //
else
    // FAIL - error handling : call KZGetError()

// [ KZPresent ]
FUNC_PRESENT funcShow;
funcShow = (FUNC_PRESENT) GetProcAddress(m_hInst, "KZPresent");
int iStart = 1;
int iCount = 20;
wchar_t *pRecSyntax = L"1.2.840.10003.5.10";
wchar_t *pElementSet = L"B";
iret = funcShow(m_nSessionID, iStart, iCount, pRecSyntax,
pElementSet);
if(iret > 0)
    // OK
else
    // FAIL

// [ KZGetBrief ]
FUNC_GET_BRIEF funcGetBrief;
funcGetBrief = (FUNC_GET_BRIEF) GetProcAddress(m_hInst, "KZGetBrief");
int iRecNo;
wchar_t szBuff[32000];
int iBuff = 32000;
for(iRecNo = 1; iRecNo <= 20; ++iRecNo)
{
```

```
    iret = funcGetBrief(m_nSessionID, iRecNo, szBuff, iBuff);
    if(iret == 1)
        // OK
    else
        // FAIL
}

// [ KZGetRecordType ]
FUNC_GET_RECORD_TYPE funcGetRecordType;
funcGetRecordType = (FUNC_GET_RECORD_TYPE) GetProcAddress(m_hInst,
"KZGetRecordType");
int iRecNo = 5;
wchar_t szBuff[32000];
int iBuff = 32000;
wchar_t *ElementSet = L"B";
iret = funcGetRecordType (m_nSessionID, iRecNo, szBuff, iBuff,
ElementSet);
if(iret == 1)
    // OK
else
    // FAIL

// [ KZAddRecord ]
FUNC_ADD_RECORD funcAddRecord;
funcAddRecord = (FUNC_ADD_RECORD) GetProcAddress(m_hInst,
"KZAddRecord");
wchar_t *pData = L"<?xml
version='1.0' ?><db_name>UDWN</db_name><job_type>INS</job_type><bib_no
>01768134</bib_no><entry_date>20020627</entry_date><entry_time>162521<
/entry_time><entry_person>
</entry_person><entry_lib>999999</entry_lib><local_system>UNICAT</loca
l_system><download_db_kind>UBIB</download_db_kind></xml>";
iret = funcAddRecord(m_nSessionID, pData);
if(iret == 1)
    // OK
else
```

```
// FAIL

// [ KZFinish ]
FUNC_CLOSE funcDis;
funcDis = (FUNC_CLOSE) GetProcAddress(m_hInst, "KZFinish");
funcDis(m_nSessionID);

// [ DLL      ]
void FreeDll()
{
    FreeLibrary(m_hInst);
    m_hInst = NULL;
}
```

[]

Bib-1 Diagnostics

```

/*****
.
*****/

```

- 3 unsupported search
- 10 Invalid format for record number (search term)
- 100 (unspecified) error
- 107 Query type not supported
- 108 Malformed query
- 109 Database unavailable database name
- 110 Operator unsupported operator
- 113 Unsupported attribute type type
- 114 Unsupported Use attribute value
- 115 Unsupported term value for Use attribute term
- 117 Unsupported Relation attribute value
- 118 Unsupported Structure attribute value
- 119 Unsupported Position attribute value
- 120 Unsupported Truncation attribute value
- 121 Unsupported Attribute Set oid
- 122 Unsupported Completeness attribute value
- 123 Unsupported attribute combination
- 125 Malformed search term
- 209 Generic sort not supported (database-specific sort only supported)
(unspecified)
- 210 Database specific sort not supported
- 218 ES: Package name already in use name
- 221 ES: extended service type not supported type
- 235 Database does not exist database name
- 239 Record syntax not supported syntax
- 1001 Malformed APDU.
- 1004 ES: Extended services not supported unless access control is in effect.
- 1008 ES: missing mandatory parameter for specified function parameter

1009 ES: Item Order, unsupported OID in itemRequest. OID

1010 Init/AC: Bad Userid

1011 Init/AC: Bad Userid and/or Password

1014 Init/AC: Authentication System error

1016 Init/AC: Blocked network address

1017 Init/AC: No databases available for specified userId

1021 Init/AC: Account has expired

1022 Init/AC: Password has expired so a new one must be supplied

1023 Init/AC: Password has been changed by an administrator so a new one must be supplied

1024 Unsupported Attribute. See note 3. an unstructured string indicating the object identifier of the attribute set id, the numeric value of the attribute type, and the numeric value of the attribute.

1025 Service not supported for this database

1027 SQL error

1040 ES: Invalid function function

1043 ES: Invalid OID for task specific parameters oid

1044 ES: Invalid action action

1049 ES: Cannot return task package -- exceeds maximum permissible size for ES response (see note 5) maximum task package size for ES response

1052 ES: Cannot process task package record -- exceeds maximum permissible record size for ES (see note 7) maximum record size for ES

1053 ES: Cannot return task package record -- exceeds maximum permissible record size for ES response (see note 8) maximum record size for ES response

1056 Attribute not supported for database attribute (oid, type, and value), and database name

1057 ES: Unsupported value of task package parameter (See Note 9) parameter and value

1071 preferredRecordSyntax not supplied