ARCHIVE:

PLEASE DO NOT DESTROY

Institute of Hydrology

1995/
097

Natural Environment Research Council

Natural
Environment
Research
Council

HYDATA

Version 4.00 (Windows)

Data manager API and database design

Oxford Scientific Software
October 1994

Last Revised August 1995

Contents

1. DATA MANAGER

1.1 Introduction
1.2 Include files (C, SAL, FORTRAN, Visual Basic)
1.3 Functions by category
1.4 Function list

2. DATABASE DESIGN

2.1 Structure
2.2 SQL scripts

1. DATA MANAGER

1.1 Introduction

The HYDATA data manager has been implemented as a Windows 3.1 DLL and is written in C. It was developed using Microsoft Visual C++ version 1.5. The name of the DLL is **HYDATA.DLL** and the name of the import library is **HYDATA.LIB**.

The library allows supports multiple database connections:

- A single application can connect to the same database more than once.
- A single application can connect to different databases at the same time.
- Multiple applications can be connected through the same DLL (and themselves have multiple connections).

The total number of connections that the DLL can support is presently set to 32. Limits on other resources may be reached before this number of connections has been made. Note that this figure applies to one PC only. If HYDATA is run over a network, this does not limit the number of users since each PC will be using its own copy of the DLL.

Each successful database connection is returned a non zero HYDATA database connection handle which must be used in all subsequent function calls relating to that connection.

The first call to the DLL loads the langauge strings into global memory. These same strings are then made available to all subsequent connections. This reduces the overhead in terms of both memory required to store the strings and load time from the database. The language that HYDATA operates in is specified in the HYDATA initialisation file **HYDATA.INI**. The "Language=" entry under the "[HYDATA]" heading may be set to English, French or Spanish. The operating langauge is therefore a property of the PC, not of the database or of the user.

All connections must start with a call to the function **HyConnect** and must be terminated by a call to **HyDisconnect**. It is important to ensure only a valid HYDATA handle is passed to functions, failure to do so will almost certainly cause a serious failure. Invalid HYDATA handles are detected by the debug version of the DLL only. The HYDATA handle is not the same as a database cursor; a single HYDATA handle maps onto multiple database cursors.

In order to allow other users to access data, it is important to call the either the function **HyCommit** or **HyRollback** as soon as possible after data are abstracted or revised.

In general all functions have return a type of BOOL. This will be TRUE if the call was successful or FALSE if there was an error. The functions **HyGetReturnStatus**, **HyGetReturnString** and **HyCopyReturnString** may be called after any function (except **HyConnect** and **HyDisconnect**) which returns a BOOL to gain more information about the outcome of the call to the previous function. (**HyConnect** and **HyDisconnect** return the status and message string on the parameter line.)

The primary test of the interface has been made using the HYDATA program manager written in SAL. A SAL include file, **HYDATA.APL**, has been written which enables SAL programs to call the DLL directly.

The parameters to functions in the data manager API have been kept a simple as possible to facilitate calls from as many programming environments as possible.

A small amount of secondary testing has been undertaken using:

- Microsoft FORTRAN v5.1 as a QuickWin application.
- Microsoft Visual Basic v3.0.
- Microsoft Visual C++ as a QuickWin application.

These three programming environments can sucessfully call the DLL but all suffer from the same (Gupta ?) bug which affects the connect and disconnect database functions. During the connect and disconnect it is necessary to send messages to the application (or window ?) by either pressing an innocuous keys (eg SHIFT) or moving the mouse over the application window. Applications written in SAL do not suffer this problem. Standard C Windows applications have not been tested.

The map manager (written as a C DLL) makes calls to the data manager. The data manager makes calls to the map manager to show selected objects. Both DLLs must therefore be present together.

1.2 Include files (C, SAL, FORTRAN, Visual Basic)

The following include files for use with different languages are provided as interface to the Data Manager DLL.

Note that the FORTRAN and Visual Basic include files only contain definitions of the functions HyConnect and HyDisconnect and therefore require more development effort.

Langauge	Include file
C	HYDATA.H
SAL	HYDATA.APL
FORTRAN	HYDATA.INC
Visual Basic	HYDATA.BAS

1.3 Functions by category

The following are a list of functions exported by the HYDATA data manager DLL. The functions are grouped according to category:

Standard message boxes

Function	**Description**
HyErrorMsg	Display the standard HYDATA error message box
HyWarnMsg	Display the standard HYDATA warning message box
HyInfoMsg	Display the standard HYDATA infromation message box
HyYesNoMsg	Display the standard HYDATA Yes/No message box

Database connection functions

Function	**Description**
HyConnect	Connect to the database
HyDisconnect	Disconnect from the database
HyAlterPassword	Changes the password of the current user

Langauge string functions

Will not be supported by VB.

Function	**Description**
* HyGetString	Gets a pointer to the requested string
HyCopyString	Copies the requested string

Error inquire functions

Function	**Description**
HyGetReturnStatus	Gets the return status for the last funcion call
* HyGetReturnMsg	Gets a pointer to the last function call return message
HyCopyRetrnMsg	Copies the last function call return message
HyTimeOut	Determines whether or not the last function failed due to a time out.

Will not be supported by VB.

General database access

Function	**Description**
HyCommit	Commit all outstanding transactions
HyRollback	Rollback all outstanding transactions
HyCount	Counts the number of entries in a HYDATA database table
HyGetNext	Gets next row for the current query
HyGetEnd	Terminate the current query and free the resources associated with it
HyInsertNext	Inserts the 2nd and subsequent rows on a multiple insertion operation
HyInsertEnd	Terminates the current insert and frees the resources associated with it
HyNextId	Gets next id which is free for a named table
HySelObj	Object selection
HyNameExists	Checks whether a name is unique
HyCountItems	Counts specific items in a table

Data access functions

Function	**Description**
HyApps	HYDATA applications
HyUnits	HYDATA measurement units
HyUpdateUnits	Updates HYDATA measurement units
HyUsers	User information control
HyObjectTypes	HYDATA object types
HyAttributes	Object attribute control
HyObjectAtt	Object type/attribute relationships
HyAttChar	Character attribute control
HyAttInt	Integer attribute control
HyAttFloat	Real number attribute control (double precision floating point)
HyAttDate	Date/time attribute control (double precision floating point)
HyAttLongChar	Long character attribute control
HYAttPic	Picture attribute control
HyPicture	Picture control
HyStationTypes	Station types
HyStations	Station definition
HyRivers	River definitions
HyRiverLocs	River locations
HyCatchments	Catchment definitions
HyCatBound	Catchment boundary definitions
HyBoundLocs	Catchment boundary points
HyMapStrings	Additional text for map annotation
HyMapLines	Additional lines and areas for map annotation
HyMapLineData	Data for additional map lines
HyDataFlags	Data description flags
HyTSInts	Time series intervals
HyTSTypes	Time series types
HyTSDef	Time series definition
HyTSReadTimes	Time series - time of data readings
HyTSData	Times series data values
HyTSExt	Times series extremes
HyGaugings	River gauging data
HySpot	River spot gaugings
HyRatDef	Rating equation definition
HyRatData	Rating equation parameters

Hydraulic structure functions

Function	**Description**
HyStructure	Hydraulic structure definition
HyStructCd	Hydraulic structure Cd definition data
HyStructData	Hydraulic structure parameter data
HyStructType	Hydraulic structure types
HyStructCdType	Hydraulic structure Cd type
HyStructPhrase	Hydraulic structure phrases
HyStructParam	Hydraulic structure parameters
HyStructFlow	Hydraulic structure flow calculation
HyStructError	Hydraulic structure flow calculation error

1.4 Function list

The following are full descriptions of functions listed in alphabetical order:

HyAlterPassword

BOOL HyAlterPassword (*hConnect, lpszOldPass, lpszNewPass*)

```
HYHAND    hConnect          /* HYDATA connection handle */
LPSTR     lpszOldPass       /* Current password *
LPSTR     lpszNewPass       /* New password *
```

The **HyAlterPassword** function allows the current user to change their database password. The password HYDATA requires from a user at log on is in fact the same as the database password.

Parameter	Description
hConnect	HYDATA connection handle associated with this connection.
lpszOldPass	The current user password (max 8 characters)
lpszNewPass	The new password (max 8 characters)

Returns

The function returns TRUE if successful or FALSE if an error occured.

Export ordinal

DLL export ordinal: 23

Comments

This function only allows the current user to change password.

Example

```
BOOL      bOK;
HYHAND    hConnect;

bOK = HyAlterPassword ( hConnect, "OLD", "NEW" );
```

HyApps

```
BOOL HyApps ( hConnect, lAppid, lpszAppName, lAppType, lpszAppExe,
              iFlag )

HYHAND   hConnect          /* HYDATA connection handle */
LPLONG   lAppId            /* Application id */
LPSTR    lpszAppName       /* Application name */
LPLONG   lAppType          /* Application type */
LPSTR    lpszAppExe        /* Name of program file to run app */
int      iFlag             /* Function control flag */
```

The **HyGetApps** function handles information about HYDATA modules and applications.

Parameter	Description
hConnect	HYDATA connection handle.
lAppId	Application id. Positive application ids are system defined, negative ids are user definable.
lpszAppName	Application name

lAppType	Application type:
	HYAPPTYPEBASIC = Basic Hydata
	HYAPPTYPEANAL = Analysis module
	HYAPPTYPEUSER = User defined application

lpszAppExe	The name of the program file to run to start the application.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first application **HYGETNEXT** - Get next application **HYUPDATE** - Update an application details **HYADD** - Add a new application **HYDELETE** - Remove an application

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 10

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and return details of the first application as ***lAppId***, ***lpszAppName***, ***lAppType*** and ***lpszAppExe***. The remaining applications are most efficiently retrieved by repeatedly calling **HyGetNext** and finally **HyGetEnd** (C and FORTRAN). For languages where the address of ***lAppId***, ***lpszAppName***, ***lAppType*** or ***lpszAppExe*** might change between calls, such as SAL, the remaining applications must be retrieved using **HyGetApps** with the **HYGETNEXT** flag. The end of the applications is signified by a return value of FALSE.

The **HYUPDATE** flag updates an existing entry for ***lpszAppName***, ***lAppType*** and ***lpszAppExe*** for a given ***lAppId***. ***lAppId***, ***lpszAppName***, ***lAppType*** and ***lpszAppExe*** must all be specified.

When the **HYADD** flag is used to add a new application. ***lAppId, lpszAppName, lAppType*** and ***lpszAppExe*** must all be specified. User defined applications must be given a negative ***lAppId*** and ***lAppType*** must be set to 3.

When the **HYDELETE** flag is used the reference to the application is removed from the database for the specified value of ***lAppType***.

Example

```
BOOL      bOK;

int       lAppId, lAppType;
char      sAppName [ 81 ];
char      sAppExe [ 81 ];
char      sTmp [ 256 ];

HYHAND    hConnect;


bOK =        HyGetApps ( hConnect, &lAppId, sAppName, &lAppType, sAppExe,
             HYGETINIT );
wsprintf ( (LPSTR) sTmp, "Id %d Name %s Type %d Exe %s", lAppId, sAppName,
             lAppType, sAppExe );


/* EITHER this code for a C or FORTRAN application */

while ( bOK )
  {
  bOK =      HyGetNext ( hConnect );
  wsprintf ( (LPSTR) sTmp, "Id %d Name %s Type %d Exe %s", lAppId, sAppName,
             lAppType, sAppExe );
  }

HyGetEnd ( hConnect );
HyCommit ( hConnect );


/* OR this code re-written in SAL for a SAL application */

while ( bOK )
  {
  bOK =      HyGetApps ( hConnect, &lAppId, sAppName, &lAppType, sAppExe,
             HYGETNEXT );
  wsprintf ( (LPSTR) sTmp, "Id %d Name %s Type %d Exe %s", lAppId, sAppName,
             lAppType, sAppExe );
  }

HyCommit ( hConnect );
```

HyAttChar

```
BOOL HyAttChar ( hConnect, lObjectTypeId, lObjectId, lAttId, lpszData,
                 iFlag )

HYHAND   hConnect          /* HYDATA connection handle */
LPLONG   lObjectTypeId     /* Object type id */
LPLONG   lObjectId         /* Object id */
LPLONG   lAttId            /* Attribute id */
LPSTR    lpszData          /* Character data */
int      iFlag             /* Function control flag */
```

The **HyAttChar** function controls character attribute data.

Parameter	Description
hConnect	HYDATA connection handle.
lObjectTypeId	Object type id
lObjectId	Object id
lAttId	Attribute id
lpszData	Attribute name
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get character attribute **HYUPDATE** - Update attribute value **HYADD** - Add a new character string **HYDELETE** - Remove a character string

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 27

Comments

Use the **HYGETINIT** flag to obtain the character string, ***lpszData***, for a specified ***lObjectTypeId***, ***lObjecttId*** and ***lAttId***.

The **HYUPDATE** flag updates an existing entry for ***lpszData***. ***lpszData***, ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** must all be specified.

When the **HYADD** flag is used to add a character string where no ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** combination exists. ***lpszData***, ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** must all be specified.

When the **HYDELETE** flag is used the entire reference to the attribute is removed from the database for the specified values of ***lObjectTypeId***, ***lObjectId*** and ***lAttId***.

Example

```
BOOL     bOK;

long     lObjectTypeId, lObjectId, lAttId;
```

```
char         sTmp [ 81 ];

HYHAND    hConnect;

/* Update character attribute data */

lObjectTypeId = 1L;
lObjectId = 2L;
lAttId = 3L;

lstrcpy ( sTmp, "New string" );

bOK = HyAttChar ( hConnect, &lObjectTypeId, &lObjectId, &lAttId, sTmp, HYUPDATE
);

HyCommit ( hConnect );
```

HyAttDate

BOOL HyAttDate (***hConnect, lObjectTypeId, lObjectId, lAttId, dtData, iFlag***)

```
HYHAND      hConnect          /* HYDATA connection handle */
LPLONG      lObjectTypeId     /* Object type id */
LPLONG      lObjectId         /* Object id */
LPLONG      lAttId            /* Attribute id */
double far * dtData           /* Date/time data */
int         iFlag             /* Function control flag */
```

The **HyAttDate** function controls date/time attribute data. These data are handled by this function a double precision variables.

Parameter	**Description**
hConnect	HYDATA connection handle.
lObjectTypeId	Object type id
lObjectId	Object id
lAttId	Attribute id
dtData	Date/time point data
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get date/time attribute **HYUPDATE** - Update date/tim value **HYADD** - Add a new date/time value **HYDELETE** - Remove a date/time value

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 30

Comments

Date/time values are handled as double precision floating point numbers. Figures before the decimal point indicate the day number; figures after the decimal point indicate the time within the day. Day numbers are relative to 30/12/1899. The time within the day is a decimal fraction (eg .5 is 12 noon). Conversion to and from this internal representation of date and time in SAL can be undertaken as follows:

(1) Declare a Date/Time constant: Date/Time: DATETIME_Base = 1899-12-30.

(2) Conversion from a SAL date (dtDate) to a SAL number (nDate) for transfer to the database:

```
Set nDate = dtDate - DATETIME_Base
```

(3) Conversion from a SAL number (nDate) to a SAL date (dtDate) for transfer from the database:

```
Set dtDate = DATETIME_Base + nDate
```

(Additional functions can be provided in the DLL to assist with this conversion if required).

Use the **HYGETINIT** flag to obtain the date/time attribute, ***dtData,*** for a specified ***lObjectTypeId, lObjecttId*** and ***lAttId***.

The **HYUPDATE** flag updates an existing entry for ***dtData***. ***dtData, lObjectTypeId, lObjecttId*** and ***lAttId*** must all be specified.

When the **HYADD** flag is used to add a date/time attribute where no ***lObjectTypeId, lObjecttId*** and ***lAttId*** combination exists. ***dtData, lObjectTypeId, lObjecttId*** and ***lAttId*** must all be specified.

When the **HYDELETE** flag is used the entire reference to the attribute is removed from the database for the specified values of ***lObjectTypeId, lObjectId*** and ***lAttId***.

Example

```
BOOL      bOK;

long      lObjectTypeId, lObjectId, lAttId;

double    dtNewDate;

HYHAND    hConnect;


/* Update an existing date/time attribute */

lObjectTypeId = 1L;
lObjectId = 2L;
lAttId = 3L;
dtNewData = 35500.5;

bOK = HyAttDate ( hConnect, &lObjectTypeId, &lObjectId, &lAttId, &dtNewData,
HYUPDATE );

HyCommit ( hConnect );
```

HyAttFloat

BOOL HyAttFloat (*hConnect, lObjectTypeId, lObjectId, lAttId, dData, iFlag*)

```
HYHAND         hConnect          /* HYDATA connection handle */
LPLONG         lObjectTypeId     /* Object type id */
LPLONG         lObjectId         /* Object id */
LPLONG         lAttId            /* Attribute id */
double far *   dData             /* Floating point data */
int            iFlag             /* Function control flag */
```

The **HyAttFloat** function controls floating point (real number) attribute data. Floating point data are double precision.

Parameter	Description
hConnect	HYDATA connection handle.
lObjectTypeId	Object type id
lObjectId	Object id
lAttId	Attribute id
dData	Floating point data
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get real attribute **HYUPDATE** - Update real value **HYADD** - Add a new real value **HYDELETE** - Remove a real value

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 29

Comments

Use the **HYGETINIT** flag to obtain the floating point attribute, ***dData***, for a specified ***lObjectTypeId***, ***lObjecttId*** and ***lAttId***.

The **HYUPDATE** flag updates an existing entry for ***dData***. ***dData***, ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** must all be specified.

When the **HYADD** flag is used to add a floating point attribute where no ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** combination exists. ***dData***, ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** must all be specified.

When the **HYDELETE** flag is used the entire reference to the attribute is removed from the database for the specified values of ***lObjectTypeId***, ***lObjectId*** and ***lAttId***.

Example

```
BOOL      bOK;
```

```
long      lObjectTypeId, lObjectId, lAttId;

double    dNewData;

HYHAND    hConnect;


/* Update an existing floating point attribute */

lObjectTypeId = 1L;
lObjectId = 2L;
lAttId = 3L;
dNewData = 1024.2048;

bOK = HyAttFloat ( hConnect, &lObjectTypeId, &lObjectId, &lAttId, &dNewData,
HYUPDATE );

HyCommit ( hConnect );
```

HyAttInt

BOOL HyAttInt (*hConnect, lObjectTypeId, lObjectId, lAttId, lData,*
iFlag)

```
HYHAND  hConnect        /* HYDATA connection handle */
LPLONG  lObjectTypeId   /* Object type id */
LPLONG  lObjectId       /* Object id */
LPLONG  lAttId          /* Attribute id */
LPLONG  lData           /* Long integer data */
int     iFlag           /* Function control flag */
```

The **HyAttInt** function controls integer attribute data. Integer data are long integers.

Parameter	Description
hConnect	HYDATA connection handle.
lObjectTypeId	Object type id
lObjectId	Object id
lAttId	Attribute id
lData	Long integer data
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get integer attribute **HYUPDATE** - Update integer value **HYADD** - Add a new integer value **HYDELETE** - Remove a integer value

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 28

Comments

Use the **HYGETINIT** flag to obtain the integer attribute, ***lData***, for a specified ***lObjectTypeId***, ***lObjecttId*** and ***lAttId***.

The **HYUPDATE** flag updates an existing entry for ***lData***. ***lData***, ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** must all be specified.

When the **HYADD** flag is used to add an integer attribute where no ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** combination exists. ***lData***, ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** must all be specified.

When the **HYDELETE** flag is used the entire reference to the attribute is removed from the database for the specified values of ***lObjectTypeId***, ***lObjectId*** and ***lAttId***.

Example

```
BOOL     bOK;
```

```
long      lObjectTypeId, lObjectId, lAttId, lNewData;

HYHAND    hConnect;


/* Update an existing integer attribute */

lObjectTypeId = 1L;
lObjectId = 2L;
lAttId = 3L;
lNewData = 1024L;

bOK  =  HyAttInt  (  hConnect,  &lObjectTypeId,  &lObjectId,  &lAttId,  &lNewData,
HYUPDATE );

HyCommit ( hConnect );
```

HyAttLongChar

```
BOOL HyAttLongChar ( hConnect, lObjectTypeId, lObjectId, lAttId, lData,
                     lBuffSize, lBytes, iFlag )

HYHAND     hConnect              /* HYDATA connection handle */
LPLONG     lObjectTypeId         /* Object type id */
LPLONG     lObjectId             /* Object id */
LPLONG     lAttId                /* Attribute id */
LPVOID     lData                 /* Pointer to data */
size_t     lBuffSize             /* Size of data buffer */
unsigned long far * lBytes       /* Number of bytes */
int        iFlag                 /* Function control flag */
```

The **HyAttLongChar** function controls long character attribute data. Long character data must be used for strings over 254 bytes. Long character data are unlimited in length and are not restricted to characeter strings in this function.

Parameter	Description
hConnect	HYDATA connection handle.
lObjectTypeId	Object type id
lObjectId	Object id
lAttId	Attribute id
lData	Pointer to the data to be transfered
lBuffSize	Size of the buffer pointed to by ***lData*** (cannot be greater than 32,767)
lBytes	If ***iFlag*** = **HYGETLONGPREP**, ***lBytes*** is the total string size in bytes If ***iFlag*** = **HYGETLONGREAD**, ***lBytes*** is the number of bytes read on the call and placed in the buffer. When a read is complete ***lBytes*** will be returned with a value of zero
iFlag	Function control flag which can take one of the following constant: **HYGETLONGPREP** - Prepare for a read **HYGETLONGREAD** - Read a block **HYGETLONGEND** - Terminate read **HYUPDATELONGPREP** - Prepare update **HYUPDATELONGADD** - Add an updated block **HYUPDATELONGEND** - End update **HYADDLONGPREP** - Prepare addition of new attribute **HYADDLONGADD** - Add an updated block **HYADDLONGEND** - End addition of new attribute **HYDELETE** - Remove a long character value

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 31

Comments

Manipulation of long character data is more complicated than single value data.

When adding a new attribute or updating an existing attribute, the long character data is supplied to the function in any number of blocks until all data are transfered to the database. The size of the block is determined by the parameter ***lBuffSize*** but cannot be greater than 32,767 bytes. The addition or update is a three stage process; preparation, adding blocks and termination. Other database calls must not be made before the operation is terminated by a call to **HyAttLongChar** with a **HYADDLONGEND** or a **HYUPDATELONGEND** flag. The operation must still be terminated, even if either of the two earlier stages resulted in an error.

When reading long character, the data are returned from the database in a series of blocks until all data are transfered. The size of the block is determined by the parameter ***lBuffSize*** but cannot be greater than 32,767 bytes. The read is a three stage process; preparation, reading blocks and termination. Other database calls must not be made before the operation is terminated by a call to **HyAttLongChar** with a **HYGETLONGEND** flag. The operation must still be terminated, even if either of the two earlier stages resulted in an error.

Getting data

First use the **HYGETLONGPREP** flag to prepare for the read for a specified ***lObjectTypeId***, ***lObjecttId*** and ***lAttId***. Note that ***lBytes*** is returned from the call and gives the total size of the character string that will be returned.

Secondly use the **HYGETLONGREAD** flag in a loop to read the data. ***lData*** and ***lBuffSize*** must be supplied. ***lBytes*** is returned from each call and gives the total number of bytes returned in the buffer. When ***lBuffSize*** is zero the read is complete.

Finaly use the **HYGETLONGEND** flag to terminate the read. No other database call must made for this connect handle until the read is terminated. The read must still be terminated, even if a failure occurs.

Update existing data/ add new data

First use the **HYUPDATELONGPREP** or **HYADDLONGPREP** flag to prepare for the write for a specified ***lObjectTypeId***, ***lObjecttId*** and ***lAttId***. "Update" should be used where the attribute has an entry on the database, "add" should be used where the particular ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** combination does not exist.

Secondly use the **HYUPDATELONGADD** or **HYADDLONGADD** flag in a loop to write the data. ***lData*** and ***lBuffSize*** must be supplied.

Finaly use the **HYUPDATELONGEND** or **HYADDLONGEND** flag to terminate the write. No other database call must made for this connect handle until the write is terminated. The write must still be terminated, even if a failure occurs.

Deleting data

When the **HYDELETE** flag is used the entire reference to the attribute is removed from the database for the specified values of ***lObjectTypeId***, ***lObjectId*** and ***lAttId***.

Example

```
BOOL      bOK;
```

```
long     lObjectTypeId, lObjectId, lAttId;
unsigned long lSize;

char     sNewDate [ 100 ];

HYHAND   hConnect;


/* Update an existing long character attribute */

lObjectTypeId = 1L;
lObjectId = 2L;
lAttId = 3L;
lstrcpy ( sNewDate, "1234567890ABCD" );

/* Prepare */

bOK = HyAttLongChar ( hConnect, &lObjectTypeId, &lObjectId, &lAttId, sNewDate,
10, &lSize, HYUPDATELONGPREP );

/* Add first block */

bOK = HyAttLongChar ( hConnect, &lObjectTypeId, &lObjectId, &lAttId, sNewDate,
10, &lSize, HYUPDATELONGADD );

/* Add second block */

bOK = HyAttLongChar ( hConnect, &lObjectTypeId, &lObjectId, &lAttId,
&sNewDate[10], 4, &lSize, HYUPDATELONGADD );

/* Terminate */

bOK = HyAttLongChar ( hConnect, &lObjectTypeId, &lObjectId, &lAttId, sNewDate,
0, &lSize, HYUPDATELONGEND );


HyCommit ( hConnect );
```

HyAttPic

```
BOOL HyAttInt ( hConnect, lObjectTypeId, lObjectId, lAttId, lPicId,
                iFlag )

HYHAND   hConnect          /* HYDATA connection handle */
LPLONG   lObjectTypeId     /* Object type id */
LPLONG   lObjectId         /* Object id */
LPLONG   lAttId            /* Attribute id */
LPLONG   lPicId            /* Picture id */
int      iFlag             /* Function control flag */
```

The **HyAttPic** function controls picture attribute data. The attribute is stored as an picture id. The actual data which defines the picture is handled with the function **HyPicture**.

Parameter	Description
hConnect	HYDATA connection handle.
lObjectTypeId	Object type id
lObjectId	Object id
lAttId	Attribute id
lPicId	Picture id. This hass the same value as ***lPicId*** used in the function **HyAttPic**
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get picture id **HYUPDATE** - Update picture id **HYADD** - Add a new picture id **HYDELETE** - Remove a picture id

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 34

Comments

Use the **HYGETINIT** flag to obtain the picture id, ***lPicId***, for a specified ***lObjectTypeId***, ***lObjecttId*** and ***lAttId***.

The **HYUPDATE** flag updates an existing entry for ***lPicId***. ***lPicId***, ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** must all be specified.

When the **HYADD** flag is used to add a picture id where no ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** combination exists. ***lPicId***, ***lObjectTypeId***, ***lObjecttId*** and ***lAttId*** must all be specified.

When the **HYDELETE** flag is used the reference to the picture id is removed from the database for the specified values of ***lObjectTypeId***, ***lObjectId*** and ***lAttId***.

Note that **HyAttPic** must be used in conjunction with **HyPicture** to enable the picture data to be retrieved, stored, modified and deleted.

Example

```
BOOL      bOK;

long      lObjectTypeId, lObjectId, lAttId, lPicId;

HYHAND    hConnect;


/* Update an existing picture id */

lObjectTypeId = 1L;
lObjectId = 2L;
lAttId = 3L;
lPicId = 23L;

bOK  =  HyAttInt  (  hConnect,  &lObjectTypeId,  &lObjectId,  &lAttId,  &lPicId,
HYUPDATE );

HyCommit ( hConnect );
```

HyAttributes

BOOL HyAttributes (*hConnect, lAttId, lpszName, iType, iFlag*)

```
HYHAND    hConnect        /* HYDATA connection handle */
LPLONG    lAttId          /* Attribute id */
LPSTR     lpszName        /* Attribute name */
LPINT     iType           /* Attribute type */
int       iFlag           /* Function control flag */
```

The **HyAttributes** function controls information concerning object type attributes. Positive attribute ids are system defined values, negative attribute ids are user defined.

Parameter	Description
hConnect	HYDATA connection handle.
lAttId	Attribute id
lpszName	Attribute name
iType	Attribute type. Can only be one of the following six values: 1 - Character (max 254 characters) 2 - Long character (unlimited length string) 3 - Integer (long) 4 - Float (double) 5 - Date 6 - Picture
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first attribute **HYGETNEXT** - Get next attribute **HYUPDATE** - Update attribute information **HYADD** - Add a new attribute **HYDELETE** - Remove an attribute

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 25

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first attribute as ***lAttId, lpszName*** and ***iType***. Data are retrieved in ascending order of ***lAttId***. Since the retrieval is complex the HyGetNext function cannot be used with this type of data. Use the **HYGETNEXT** flag on **HyAttributes** to get all remaining data.

When the **HYUPDATE** flag is used ***lpszName*** is changed for the attribute id specified in ***lAttId***. (***iType*** is not used.) Applications must not change system defined attribute names (ie ***iAttId*** MUST be negative).

When the **HYADD** flag is used a new attribute is added to the database. ***lAttId***, ***lpszName*** and ***iType*** are all required for this type of function call. Applications must only add user defined attributes (ie ***iAttId*** MUST be negative).

When the **HYDELETE** flag is used the attribute is removed from the database together with any data held for that attribute in the attribute data tables. ***lAttId*** must be specified for this call. (***iType*** is returned as the type.)

Example

```
BOOL     bOK;

int      iType;

long     lAttId;

HYHAND   hConnect;


/* Add a new attribute */

lAttId = -3L;
iType = 3;

bOK = HyAttribute ( hConnect, &lAttId, "New att", &iType, HYADD );

HyCommit ( hConnect );
```

HyBoundLocs

```
BOOL HyBoundLocs ( hConnect, lBoundLocId, fX, fY, iFlag )

HYHAND      hConnect          /* HYDATA connection handle */
LPLONG      lBoundLocId       /* Boundary location id */
double far * fX               /* Map x co-ordinate */
double far * fY               /* Map y co-ordinate */
int         iFlag             /* Function control flag */
```

The **HyBoundLocs** function controls information concerning boundary points for all catchments.

Parameter	Description
hConnect	HYDATA connection handle.
lBoundLocId	Boundary location id.
fX	Catchment boundary location x co-ordinate
fY	Catchment boundary location y co-ordinate
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first boundary location point **HYGETNEXT** - Get next boundary location point **HYUPDATE** - Update boundary location point **HYADD** - Add a new boundary point **HYDELETE** - Remove a boundary point

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 41

Comments

This function handles the X-Y co-ordinates of all catchment boundary points. Some of these points may be used by more than one catchment. Use the function **HyCatBounds** to determine which of the boundary points define a particular catchment.

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first boundary location point as ***lBoundLocId***, ***fX*** and ***fY*** for a specified catchment ***lCatId***. The function **HyGetNext** can be used with this type of data to get the remaining boundary points.

Use the **HYUPDATE** flag to change ***fX*** and ***fY*** for a specified ***lBoundLocId***.

When the **HYADD** flag is used a new boundary point is added to the database. ***lBoundLocId***, ***fX*** and ***fY*** are all required for this type of function call.

When the **HYDELETE** flag is used the catchment boundary point is removed from the database. ***lBoundLocId*** must be specified for this call. Care must be taken in removing boundary points since individual points may be used by more than one catchment.

Example

```
BOOL      bOK;

long      lBoundLocId;

double    fX, fY;

HYHAND    hConnect;


/* Add a new catchment boundary point */

lBoundLocId = 10L;
fX = 1002.34;
fY = 23494.45;

bOK = HyBoundLocs ( hConnect, &BoundLocId, &fX, &fY, HYADD );

HyCommit ( hConnect );
```

HyCatBounds

```
BOOL HyCatBounds ( hConnect, lCatId, lBoundLocId, lNextId,iFlag )

HYHAND    hConnect           /* HYDATA connection handle */
LPLONG    lCatId             /* Catchment id */
LPLONG    lBoundLocId        /* Boundary location id */
LPLONG    lNextId            /* Next boundary location id */
int       iFlag              /* Function control flag */
```

The **HyCatBound** function controls information concerning river catchments boundaries.

Parameter	**Description**
hConnect	HYDATA connection handle.
lCatId	Catchment id.
lBoundLocId	Boundary location id.
lNextId	Next boundary location id. ***lBoundLocId*** and ***lNextId*** form a connected polygon. The start of the catchment boundary (where the catchment crosses the river) is determined by the function **HyCatchments**.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first boundary point **HYGETNEXT** - Get next boundary point **HYADD** - Add a new boundary point **HYDELETE** - Remove a boundary

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 40

Comments

This function handles the indivual points which make up a catchment's boundary. The same boundary data point may be in use by more than one catchment. The function **HyBoundLocs** is used to add, update and delete individual boundary points.

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first boundary point as ***lCatId, lBoundLocId, lNextId,***. The function **HyGetNext** can be used with this type of data to get the remaining boundary points.

When the **HYADD** flag is used a new catchment boundary point is added to the database. ***lCatId, lBoundLocId*** and ***lNextId*** are all required for this type of function call. Note that ***fX*** and ***fY*** must be added using the fucntion **HyBoundLocs** since the same boundary point may be used by more than one catchment.

When the **HYDELETE** flag is used the catchment boundary definition is removed from the database. ***lCatId*** must be specified for this call. Note that the boundary location points themselves are not removed since they may be used by another catchment. Boundary points must be removed by the fucntion **HyBoundLocs**.

Example

```
BOOL      bOK;

long      lCatId, lBoundLocId, lNextId;

HYHAND    hConnect;


/* Add a new catchment boundary definition point */

lCatId = 43L;
lBoundLocId = 10L;
lNextId = 11L;

bOK = HyCatBound ( hConnect, &lCatId, &BoundLocId, &NextId, HYADD );

HyCommit ( hConnect );
```

HyCatchments

```
BOOL HyCatchments ( hConnect, lId, sName, lParentId, lStationId,
                    lRiverLocId, lBoundLocId, iFlag )

HYHAND   hConnect        /* HYDATA connection handle */
LPLONG   lId             /* Catchment id */
LPSTR    sName           /* Catchment name */
LPLONG   lParentId       /* Catchment id of parent catchment */
LPLONG   lStationId      /* Station id of main gauging station */
LPLONG   lRiverLocId     /* River loc. id of start of catchemnt */
LPLONG   lBoundLocId     /* Boundary location id */
int      iFlag           /* Function control flag */
```

The **HyCatchments** function controls information concerning river catchments.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Catchment id.
sName	Name of catchment
lParentId	Catchment id of the parent catchment. If there is no parent catchment this is zero. Catchments can be nested to any depth.
lStationId	Station id of the main catchment gauging station.
lRiverLocId	River location id where the catchment boundary crosses the river.
lBoundLocId	Boundary location id where the boundary crosses the river.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first catchment **HYGETNEXT** - Get next catchment **HYUPDATE** - Update catchment details **HYADD** - Add a new catchment **HYDELETE** - Remove a catchment

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 39

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first catchment as ***lId***, ***sName***, ***lParentId***, ***lStationId***, ***lRiverLocId*** and ***lBoundLocId***. Data are retrieved in ascending order of ***lId***. The function **HyGetNext** can be used with this type of data to get the remaining catchments.

When the **HYUPDATE** flag is used ***sName***, ***lParentId***, ***lStationId***, ***lRiverLocId***, and ***lBoundLocId*** are changed for the catchment id specified in ***lId***.

When the **HYADD** flag is used a new catchment is added to the database. ***lId, sName, lParentId, lStationId, lRiverLocId*** and ***lBoundLocId*** are all required for this type of function call.

When the **HYDELETE** flag is used the catchment is removed from the database. ***lId*** must be specified for this call.

Example

```
BOOL      bOK;

long      lId, lParentId, lStationId, lRiverLocId, lBoundLocId;

HYHAND    hConnect;


/* Add a new catchment */

lId = 43L;
lParentId = 0L;
lStationId = 105L;
lRiverLocId = 1023L;
lBoundLocId = 23412L;

bOK = HyCatchments ( hConnect, &lId, "New catchment", &lParentId, &lStationId,
&lRiverLocId, &lBoundLocId, HYADD );

HyCommit ( hConnect );
```

HyCommit

```
BOOL HyCommit ( hConnect )

HYHAND    hConnect          /* HYDATA connection handle */
```

The **HyCommit** function commits all transactions outstanding for the application and allows other users access to data which has been modified.

Parameter	Description
hConnect	HYDATA connection handle associated with this connection.

Returns

The function returns TRUE if successful or FALSE if an error occured.

Export ordinal

DLL export ordinal: 20

Comments

This function or **HyRollback** should be used as soon as possible after data are abstracted or altered to enable access by other users.

Example

```
BOOL      bOK;
HYHAND    hConnect;

if ( HyUpdateUnits ( hConnect, 4L, "New name", 25.4, 0.2, 3 ) )
  HyCommit ( hConnect );
else
  HyRollback ( hConnect );
```

HyConnect

```
BOOL HyConnect ( iAppid, lpszDatabase, lpszUser, lpszPassword,
                 bRecover, iUserId, iAuth, bInform, &iStatus,
                 lpszReturn, &hConnect )

int       iAppId        /* Application id */
LPSTR     lpszDatabase  /* Name of the database */
LPSTR     lpszUser      /* Name of the user */
LPSTR     lpszPassword  /* User password */
BOOL      bRecover      /* Database recovery on/off */
LPINT     iUserId       /* User id */
LPINT     iAuth         /* User authority (privilege level) */
BOOL      bInform       /* Display error message box on/off */
LPINT     iStatus       /* Return status code */
LPSTR     lpszReturn    /* Return status string */
LPHYHAND  hConnect      /* HYDATA connection handle */
```

The **HyConnect** function must be called before any other functions in the DLL for a given connect. The function returns a unique handle in the variable ***hConnect*** which must be used in all subsequent calls for this connection. A connection must be terminated by a call to **HyDisconnect**. The first connect after the DLL is called loads the language strings into memory which are then used by all subsequent connects.

Parameter	Description
iAppId	Application id. Currently assigned application ids are: <u>ID</u> <u>Application</u> 1 Program manager 2 Data manager 3 Map manager 4 Graph manager 5 Time series editor
lpszDatabase	The name of the database to connect
lpszUser	The name of the user
lpszPassword	The user password
bRecover	Set to TRUE if the database is connected with reocvery set ON or to FALSE to connect with recovery set OFF. For the implications of this parameter see GUPTA documentation. In general and unless you are sure of the implications recovery should be on and the parameter set to TRUE.
iUserId	User id. A unique number assigned to each user.
iAuth	User authority (privilege level). After a sucessful connect this will be an integer in the range 1 to 3. The application must ensure that the restrictions relating to lower privilege levels are carried out.
bInform	If set to TRUE a message box is displayed by the DLL itself when a failure occurs. It may be useful to set this parameter to TRUE when debugging an application. It should always be set to FALSE for the release version
iStatus	This variable will be set to a non zero status code when this function fails

lpszReturn	This user defined string space is filled with a string which indicates the outcome of the call to this function. This string should be at least 384 bytes long.
hConnect	If the function makes a successful connection to the database this will contain the non zero unique HYDATA connection handle associated with this connection.

Returns

The return value is TRUE if the function is successful or FALSE otherwise.

Export ordinal

DLL export ordinal: 2

Comments

Example

```
int       iret, iStatus, iUserId, iAuth;
HYHAND    hConnect;
char      sBuff [ 384 ];


iRet =       HyConnect ( 2, "HYDATA", "HYDATA", "HYDATA", TRUE, &iUserId, &iAuth,
             TRUE, &iStatus, sBuff, &hConnect );
```

HyCopyReturnMsg

```
void HyGetReturnMsg ( hConnect, lpszBuff )

HYHAND   hConnect          /* HYDATA connection handle */
LPSTR    lpszBuff          /* Pointer to string to hold message */
```

The **HyCopReturnMsg** function copies the string associated with the the return status of the last function call into the string specified on the parameter line. This function must not be used after calls to **HyConnect** or **HyDisconnect**. These two functions provide the same information on the command line.

Parameter	Description
hConnect	HYDATA connection handle associated with this connection.
lpszBuff	A pointer to a previously assigned string. The maximum size of string is 255 bytes including the null terminating character

Returns

There is no return value from this function

Export ordinal

DLL export ordinal: 8

Comments

Example

```
char      sTmp [ 300 ];
char      sTmpA [ 350 ];

HYHAND    hConnect;

HyCopyReturnMsg ( hConnect, (LPSTR) sTmp )
iRet =      wsprintf ( (LPSTR) sTmpA, "The return message is %s", sTmp );
```

HyCopyString

```
void HyCopyString ( hConnect, lAppId, iStringTypeId, iStringId,
                    lpszBuff )

HYHAND   hConnect          /* HYDATA connection handle */
int      lAppId            /* Application id */
int      iStringTypeId     /* String type id */
long     iStringId         /* String id */
LPSTR    lpszBuff          /* User defined string space */
```

The **HyCopyString** function retrieves the requested string for the langauge which defined in **HYDATA.INI** and copies that string into the string provided by the caller.

Parameter	Description
hConnect	HYDATA connection handle.
lAppid	Application id (see HyConnect). One application may retrieve strings belonging to other applications. Generic strings are retrieved by setting ***lAppId*** to zero.
iStringTypeId	Set to 1 (one) for error message strings. Other string types are application dependent.
iStringId	String id for the specified application and string type.
lpszBuff	A pointer to a previously assigned string. The maximum size of string is 255 bytes including the null terminating character

Returns

There is no return value.

Export ordinal

DLL export ordinal: 5

Comments

Strings are pre-loaded into global memory on the first successful connect that the DLL makes. All subsequent connections, whether to the same data base or not, use the same set of strings.

Use this function instead of HyGetString when calling from a non C application (eg SAL, FORTRAN, Visual Basic). Ensure that *lpszBuff* points to a string of at least 255 bytes in length.

Example

```
char     sBuff [ 255 ];
char     sTmp [ 300 ];

HYHAND   hConnect;


            HyCopyString ( hConnect, 2, 1, 4L, (LPSTR) sBuff )
```

```
wsprintf ( (LPSTR) sTmp, "The string is %s", sBuff );
```

HyCount

```
BOOL HyCount ( hConnect, lpszTableName, lTotal )

HYHAND    hConnect           /* HYDATA connection handle */
LPSTR     lpszTableName      /* Table name */
LPLONG    lTotal             /* Number of entries */
```

The **HyCount** function returns the number of entries (rows) in a database table.

Parameter	Description
hConnect	HYDATA connection handle.
lpszTableName	A pointer to a null terminated string containing the name of the database table.
lTotal	Set on exit to the total number of entries (rows) in the table.

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 9

Comments

Example

```
long      lTotal;

HYHAND    hConnect;


iRet =       HyCount ( hConnect, (LPSTR) "STATION", &lTotal );
```

HyCountItems

```
BOOL HyCount ( hConnect, lId, lpszTableName, lpszColumn, lTotal )

HYHAND   hConnect          /* HYDATA connection handle */
LPLONG   lId               /* Numeric value of lpszColumnName for search */
LPSTR    lpszTableName     /* Table name */
LPSTR    lpszColumnName    /* Column name for restricted serach */
LPLONG   lTotal            /* Number of entries found */
```

The **HyCountItems** function returns the number of entries (rows) in a database table for a specific numeric (long) value in a named column of thta table.

Parameter	Description
hConnect	HYDATA connection handle.
lId	The numeric value as type long in column ***lpszColumnName*** for the restricted search
lpszTableName	A pointer to a null terminated string containing the name of the database table.
lpszTableName	A pointer to a null terminated string containing the name of the database column containing ***lId***.
lTotal	Set on exit to the total number of entries (rows) in the table meeting the condition.

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 70

Comments

Example

```
long      lId, lTotal;

HYHAND    hConnect;

lId = 14;

/* Find the number of hydraulic structures at station id 14 */

iRet =      HyCount ( hConnect, &lId, (LPSTR) "STRUCTURE", (LPSTR) "STATION_ID",
            &lTotal );
```

HyDataFlags

```
BOOL HyDataFlags ( hConnect, lId, lpszName, iFlag )

HYHAND   hConnect          /* HYDATA connection handle */
LPLONG   lId               /* Data flag id */
LPSTR    lpszName          /* Data flag name */
int      iFlag             /* Function control flag */
```

The **HyDataFlagss** function controls information concerning data flag definition. Data flags are used to add descriptive information to individual data items (eg. "Missing", "Interpolated").

Parameter	Description
hConnect	HYDATA connection handle.
lId	Data flag id. Positive ids are reserved for system defined flags. Negative ids are for the use of individual users.
lpszName	Data flag name
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first data flag **HYGETNEXT** - Get next data flag **HYUPDATE** - Update data flag name **HYADD** - Add a new data flag **HYDELETE** - Remove a data flag

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 45

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first data flag as ***lId*** and ***lpszName***. Data are retrieved in ascending order of ***lId***. The function **HyGetNext** cannot be used with this type of data to get the remaining data flags due to the complex nature of the retrieval.

When the **HYUPDATE** flag is used ***lpszName*** is changed for the data flag id specified in ***lId***. Only the names of user defined data flags (with negative ids) should be changed.

When the **HYADD** flag is used a new data flag is added to the database. ***lId*** and ***lpszName*** are both required for this type of function call. Users may only add data flags with a negative id.

When the **HYDELETE** flag is used the data flag be removed from the database. ***lId*** must be specified for this call. Any data associated with the data flag is not deleted; it is the responsibility of the application to make sure that it is. Users may only delete data flags with a negative id.

Example

```
BOOL      bOK;

long      lId;

HYHAND    hConnect;


/* Add a new data flag */

lId = -4L;

bOK = HyDataFlags ( hConnect, &lId, "Very poor", HYADD );

HyCommit ( hConnect );
```

HyDisconnect

```
BOOL HyDisconnect ( hConnect, &iStatus, lpszReturn )

HYHAND   hConnect          /* HYDATA connection handle */
LPINT    iStatus           /* Return status code */
LPSTR    lpszReturn        /* Return status string */
```

The **HyDisconnect** function must be called by the application to terminate the connection to the database.

Parameter	Description
hConnect	HYDATA connection handle associated with this connection.
iStatus	This variable will be set to a non zero status code when this function fails
lpszReturn	This user defined string space is filled with a string which indicates the outcome of the call to this function. This string should be at least 384 bytes long.

Returns

The return value is TRUE if the function is successful or FALSE otherwise.

Export ordinal

DLL export ordinal: 3

Comments

HyDisconnect frees resources associated with the connection and disconnects all cursors from the database. The handle ***hConnect*** must no be used to call any other database functions after a disconnect unless another connect has been performed.

Example

```
int        iret, iStatus;

char       sBuff [ 384 ];

HYHAND     hConnect;


iRet =        HyDisconnect ( hConnect, &iStatus, sBuff );
```

HyErrorMsg

```
void HyErrorMsg ( hConnect, hwndParent, lAppId, lStringId,
                  lpszExtraInfo )

HYHAND   hConnect          /* HYDATA connection handle */
HWND     hwndParent        /* Parent window handle */
int      lAppId            /* Application id */
long     lStringId         /* Error number (String id) */
LPSTR    lpszExtraInfo     /* Additional error information */
```

The **HyErrorMsg** function displays the standard HYDATA error message box.

Parameter	Description
hConnect	HYDATA connection handle associated with this connection.
hwndParent	The parent window handle for the message box. Use (HWND) 0 if no parent window.
lAppId	Application id. Use (int) 0 if a generic error message is to be displayed.
lStringId	String id (error number).
lpszExtraInfo	A pointer to a string containing additional information to be displayed in the message box to assist the user with understanding the error. Pass a a string of zero length if no additional information is to be displayed

Returns

There is no return value.

Export ordinal

DLL export ordinal: 16

Comments

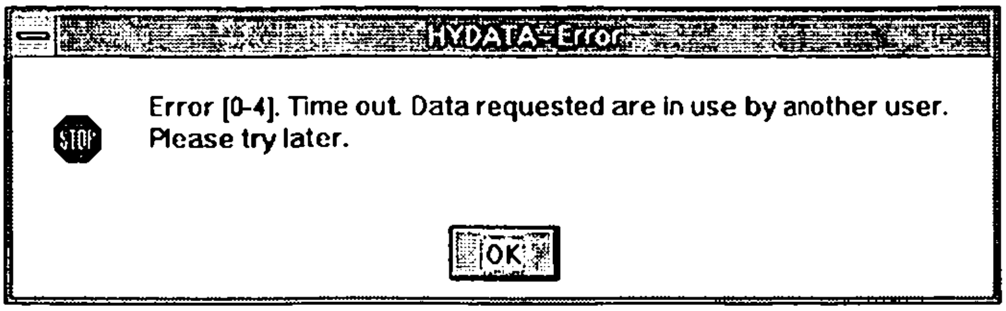
This function should be used to display all HYDATA error messages so that they appear consistent to the user. Error message strings are added to the database using the HYLANG utility program. Error message strings always have a string type of 1 for all applications.

Example

The following is an example of the standard HYDATA error message box with the two part error identification code. Application id zero is a generic error, not specific to any one application.

```
HYHAND    hConnect;

HyErrorMsg ( hConnect, (HWND) 0, 0, 4L, "" );
```



Error [0-4]. Time out. Data requested are in use by another user. Please try later.

Application id 0, Error message number 4

HyGaugings

BOOL HyGaugings (*hConnect, lpszTable, fReadTime, fLevel, fFlow, fVel, lpszRatName, lpszComments, iFlag*)

```
HYHAND       hConnect        /* HYDATA connection handle */
LPSTR        lpszTable       /* Gauging time series table name */
double far * fReadTime       /* Date and time of gauging */
double far * fLevel          /* Water level */
double far * fFlow           /* Total discharge */
double far * fVel            /* Mean velocity */
LPSTR        lpszRatName     /* Rating name */
LPSTR        lpszComments    /* Comments */
int          iFlag           /* Function control flag */
```

The **HyGaugings** function controls information concerning river gaugings.

Parameter	Description
hConnect	HYDATA connection handle.
lpszTable	Gauging time series table name
fReadTime	Date and time the gauging was undertaken
fLevel	Water level at time of gauging
fFlow	Total discharge of the gauging
fVel	Mean velocity of the gauging (Q/A)
lpszRatName	Rating name that the gauging has been associated with (? for not associated, + used in all ratings)
lpszComments	Comments on the gauging
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first gauging **HYGETNEXT** - Get next gauging **HYUPDATE** - Update gauging **HYADD** - Add a new gauging **HYDELETE** - Remove a gauging **HYDELETEALL** - Remove all gaugings in the table **HYINSERTINIT** - Insert first gauging in a block **HYINSERTNEXT** - Insert next gauging in a block

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 53

Comments

Use the **HYGETINIT** flag on the first call to prepare the query for a specified ***lpszTable*** and the function return details of the first gauging as ***fReadTime***, ***fLevel***, ***fFlow***, ***lpszRatName*** and ***lpszComments***. Data are retrieved in ascending order of ***fReadTime***. Use the **HyGetNext** function to get all remaining data.

When the **HYUPDATE** flag is used ***fLevel***, ***fFlow***, ***fVel***, ***lpszRatName*** and ***lpszComments*** are changed for the specified ***fReadTime***.

When the **HYADD** flag is used a new gauging is added to the database. All parameters are required for this type of function call.

When the **HYDELETE** flag is used the gauging identified by ***fReadTime*** is removed from the database. The **HYDELETEALL** flags deletes all gaugings in the time series.

A block of gaugings can be inserted more efficiently than by repeated use of the **HYADD** flag by using the **HYINSERTINIT** flag for the first gauging and then calling the function **HyInsertNext** for all subsequent gaugings. All parameters must be supplied. For languages where the address of function parameters changes (eg SAL), the **HYINSERTNEXT** flag must be used with the function **HyGaugings** (with all parameters supplied) rather than using the faster **HyInsertNext** function. The **HyInsertEnd** function must be called after the final insert to free resources associated with the insert.

Example

```
BOOL      bOK;

long      lpszRatName;

double    fReadTime, fLevel, fFlow, fVel;

char      sTable [ 81 ], sComments [ 257 ];

HYHAND    hConnect;


/* Add a new gauging */

lstrcpy ( sTable, "GG23" );
lpszRatName = 0L;

fLevel = 10.34;
fFlow = 20.34;
fVel = 0.898;

fReadTime = 34355.5;

lstrcpy ( sComment, "New gauging" );

bOK = HyGaugings ( hConnect, sTable, &fLevel, &fFlow, &fVel, &lpszRatName,
sComments, HYADD );

HyCommit ( hConnect );
```

HyGetEnd

BOOL HyGetEnd (*hConnect*)

```
HYHAND    hConnect           /* HYDATA connection handle */
```

The **HyGetEnd** function terminates the currently active 'get' and frees the resources associated with the activity. This function must be called after the final **HyGetNext** in a data retrieval.

Parameter	Description
hConnect	HYDATA connection handle.

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 11

Comments

This function must be called after the final call to **HyGetNext** to free resources associated with the retrieval.

Example

See the **HyGetApps** function.

HyGetNext

```
BOOL HyGetNext ( hConnect )

HYHAND   hConnect          /* HYDATA connection handle */
```

The **HyGetNext** function gets the next row from the database table after the initial query has been set up.

Parameter	Description
hConnect	HYDATA connection handle.

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 12

Comments

This function should be used with langauges such as C or FORTRAN where the address of the receive variables does not change between calls. If this is not the case (eg SAL) the flag **HYGETNEXT** should be used with the original query function. Use **HyGetNext** if possible since this is the most efficient and fastest retrieval method.

A return of FALSE indicates the end of dataset has been reached.

Example

See the **HyGetApps** function.

HyGetReturnStatus

int HyGetReturnStatus (*hConnect*)

```
HYHAND    hConnect          /* HYDATA connection handle */
```

The **HyGetReturnStatus** function returns the status number associated with the previous function call to the DLL. The value of this number determines the type of error. This function must not be used after calls to **HyConnect** or **HyDisconnect**. These two functions provide the same information on the command line.

Parameter	Description
hConnect	HYDATA connection handle associated with this connection.

Returns

The return value is the status value.

Export ordinal

DLL export ordinal: 6

Comments

Status values are:

Status	Description
0	No error

Example

```
int       iStatus;

HYHAND    hConnect;


iStatus =   HyGetReturnStatus ( hConnect );
```

HyGetReturnMsg

LPSTR HyGetReturnMsg (*hConnect*)

```
HYHAND    hConnect            /* HYDATA connection handle */
```

The **HyGetReturnMsg** function returns a pointer to a null terminated string associated with the the return status of the last function call. This function must not be used after calls to **HyConnect** or **HyDisconnect**. These two functions provide the same information on the command line.

Parameter	Description
hConnect	HYDATA connection handle associated with this connection.

Returns

The return value is a pointer to the message string.

Export ordinal

DLL export ordinal: 7

Comments

Example

```
char      sTmp [ 300 ];

HYHAND    hConnect;


iRet =       wsprintf ( (LPSTR) sTmp, "The return message is %s", HyGetReturnMsg
             ( hConnect ) );
```

HyGetString

LPSTR HyGetString (*hConnect, lAppId, iStringTypeId, iStringId*)

```
HYHAND   hConnect          /* HYDATA connection handle */
int      lAppId            /* Application id */
int      iStringTypeId     /* String type id */
long     iStringId         /* String id */
```

The **HyGetString** function returns a pointer to the requested string for the langauge which defined in **HYDATA.INI**.

Parameter	Description
hConnect	HYDATA connection handle.
lAppId	Application id (see HyConnect). One application may retrieve strings belonging to other applications. Generic strings are retrieved by setting ***lAppId*** to zero.
iStringTypeId	Set to 1 (one) for error message strings. Other string types are application dependent.
iStringId	String id for the specified application and string type.

Returns

The return value is a pointer to the requested string. If the string specified cannot be found a pointer to the default string is returned.

Export ordinal

DLL export ordinal: 4

Comments

Strings are pre-loaded into global memory on the first successful connect that the DLL makes. All subsequent connections, whether to the same data base or not, use the same set of strings.

The maximum string length is 255 characters (including the terminating null).

Example

```
char      sTmp [ 300 ];

HYHAND    hConnect;


iRet =       wsprintf ( (LPSTR) sTmp, "The string is %s", HyGetString ( hConnect,
             2, 1, 4L ) );
```

HyInfoMsg

```
void HyInfoMsg ( hConnect, hwndParent, lpszInfo )

HYHAND    hConnect           /* HYDATA connection handle */
HWND      hwndParent         /* Parent window handle */
LPSTR     lpszInfo           /* Information to display */
```

The **HyInfoMsg** function displays the standard HYDATA information message box.

Parameter	Description
hConnect	HYDATA connection handle associated with this connection.
hwndParent	The parent window handle for the message box. Use (HWND) 0 if no parent window.
lpszInfo	A pointer to a string containing information to be displayed in the message box

Returns

There is no return value.

Export ordinal

DLL export ordinal: 18

Comments

This function should be used to display all HYDATA information messages so that they appear consistent to the user.

Example

```
HYHAND    hConnect;

/* Display string 15 (string type 3) for application id 6 */

HyInfoMsg ( hConnect, (HWND) 0, HyGetString ( hConnect, 6, 3, 15L ) );
```

HyInsertEnd

```
BOOL HyInsertEnd ( hConnect )

HYHAND    hConnect          /* HYDATA connection handle */
```

The **HyInsertEnd** function terminates the currently active insert and frees the resources associated with the activity. This function must be called after the final **HyInsertNext** in a data insertion.

Parameter	Description
hConnect	HYDATA connection handle.

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 67

Comments

This function must be called after the final call to **HyInsertNext** to free resources associated with the insertion.

Example

See the **HyGaugings** function.

HyInsertNext

```
BOOL HyInsertNext ( hConnect )

HYHAND    hConnect          /* HYDATA connection handle */
```

The **HyInsertNext** function inserts the next row of data into a database table after the initial insert has been set up.

Parameter	Description
hConnect	HYDATA connection handle.

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 66

Comments

This function should be used with langauges such as C or FORTRAN where the address of the receive variables does not change between calls. If this is not the case (eg SAL) the flag **HYINSERTNEXT** should be used with the original insert function. Use **HyInsertNext** if possible since this is the most efficient and fastest insert method.

Example

See the **HyGaugings** function.

HyMapLineData -

BOOL HyMapLineData (*hConnect, lId, lOrderNo, fX, fY, iFlag*)

```
HYHAND      hConnect          /* HYDATA connection handle */
LPLONG      lId               /* Map line id */
LPLONG      lOrderNo          /* Plot order number */
double far * fX               /* Map x co-ordinate */
double far * fY               /* Map y co-ordinate */
int         iFlag             /* Function control flag */
```

The **HyMapLineData** function controls information concerning the data for plotting lines used for annotation the map. The line definition is handled by the function **HyMapLines**.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Map line id.
lOrderNo	Plot order number for each segment of the polyline. Line segments will be returned in this order.
fX	Map x co-ordinate.
fY	Map y co-ordinate
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first map line data point **HYGETNEXT** - Get next map line data point **HYUPDATE** - Update map line data point **HYADD** - Add a new map line data point **HYDELETE** - Remove a map line data point

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 44

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first map line data point as ***lId, lOrderNo, fX*** and ***fY***. Data are retrieved in ascending order of ***lId*** and then for ***lOrderNo*** within each ***lId***. The function **HyGetNext** can be used with this type of data to get the remaining map line data points.

When the **HYUPDATE** flag is used ***fX*** and ***fY*** are changed for the map line data point specified by ***lId*** and ***lOrderNo***.

When the **HYADD** flag is used a new map line data point is added to the database. ***lId, lOrderNo, fX*** and ***fY*** are all required for this type of function call.

When the **HYDELETE** flag is used the map line data point removed from the database. ***lId*** and ***lOrderNo*** must be specified for this call.

Example

```
BOOL      bOK;

long      lId, lOrderNo;

double    fX, fY;

HYHAND    hConnect;


/* Add a new map line data point */

lId = 435L;
lOrderNo = 1L;
fX = 1.0;
fY = 2.0;

bOK = HyMapLineData ( hConnect, &lId, &lOrderNo, &fX, &fY, HYADD );

HyCommit ( hConnect );
```

HyMapLines

BOOL HyMapLines (***hConnect, lId, sName, lVisLev, fThick, lStyle, lColourId, lFillStyleId, iFlag***)

```
HYHAND     hConnect            /* HYDATA connection handle */
LPLONG     lId                 /* Map line id */
LPSTR      sName               /* Name of line */
LPLONG     lVisLev             /* Visibility level */
double far *  fThickness       /* Line thickness */
LPLONG     lStyleId            /* Style id */
LPLONG     lColourId           /* Colour id */
LPLONG     lFillStyleId        /* Fill style id */
int        iFlag               /* Function control flag */
```

The **HyMapLines** function controls information concerning lines used for annotation the map. The data for drawing each line are handled by the function **HyMapLineData**.

Parameter	**Description**
hConnect	HYDATA connection handle.
lId	Map line id.
sName	Name identifier for line.
lVisLev	Visibility level of string (>=100). 100 = always visible; 200 = only visible when map zoom is 200%, etc.
fThick	Set to zero to draw as pattern (dashed). Set to 1, 2, 3, 4 or 5 for thick solid lines.
lStyleId	Set to 0, 1, 2, 3, or 4. Zero is solid; other numbers different pattern dashed lines. This parameter is only used if ***fThick*** is set to zero.
lColourId	Colour id for drawing the text string
lFillStyleId	Set to zero if area bounded by line is not to be filled. A positive value indicates the fill pattern.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first map line **HYGETNEXT** - Get next map line **HYUPDATE** - Update map line details **HYADD** - Add a new map line **HYDELETE** - Remove a map line

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 43

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first map line as ***lId, sName, lVisLev, fThick, lStyleId, lColourId*** and ***lFillStyleId***. Data are retrieved in ascending order of ***lId***. The function **HyGetNext** can be used with this type of data to get the remaining map lines.

When the **HYUPDATE** flag is used ***sName, lVisLev, fThick, lStyleId, lColourId*** and ***lFillStyleId*** are changed for the map line id specified in ***lId***.

When the **HYADD** flag is used a new map line is added to the database. ***lId, sName, lVisLev, fThick, lStyleId, lColourId*** and ***lFillStyleId*** are all required for this type of function call.

When the **HYDELETE** flag is used the map line is removed from the database. ***lId*** must be specified for this call. The map line data for the whole line is also removed from the database.

Example

```
BOOL      bOK;

long      lId, lVisLev, lStyleId, lColourId, lFillStyleId;

double    fThick;

HYHAND    hConnect;


/* Add a new map line description */

lId = 435L;
lVisLev = 100L;
lStyleId = 1L;
lColourId = 2L;
fThick = 0.0;
lFillStyleId = 0L;

bOK = HyMapLines ( hConnect, &lId, "Line1", &lVisLev, &fThick, &lStyleId,
&lColourId, &lFillStyleId, HYADD );

HyCommit ( hConnect );
```

HyMapStrings

```
BOOL HyMapStrings ( hConnect, lId, sText, lVisLev, fX, fY, fWidth,
                    fAngle, lSymbolId, lColourId, iFlag )

HYHAND      hConnect         /* HYDATA connection handle */
LPLONG      lId              /* Map string id */
LPSTR       sText            /* Text string to draw */
LPLONG      lVisLev          /* Visibility level */
double far * fX              /* X co-ordinate */
double far * fY              /* Y co-ordinate */
double far * fWidth          /* Character width */
double far * fAngle          /* Draw angle */
LPLONG      lSymbolId        /* Symbol id */
LPLONG      lColourId        /* Colour id */
int         iFlag            /* Function control flag */
```

The **HyMapStrings** function controls information concerning character strings used for annotation the map.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Map string id.
sText	Text of string to appear on map.
lVisLev	Visibility level of string (>=100). 100 = always visible; 200 = only visible when map zoom is 200%, etc.
fX	Map X co-ordinate for position of symbol (or start of text if no symbol is drawn).
fY	Map Y co-ordinate for position of symbol (or start of text if no symbol is drawn).
fWidth	Width of characters in map internal units.
fAngle	Angle of draw for characters. 0 = horizontal increasing anti-clockwise
lSymbolId	Id of the symbol to draw at the start of text. The symbol is plotted at ***fX, fY***. Set ***lSymobolId*** to -1 to inhibit symbol draw.
lColourId	Colour id for drawing the text string
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first map string **HYGETNEXT** - Get next map string **HYUPDATE** - Update map string details **HYADD** - Add a new map string **HYDELETE** - Remove a map string

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 42

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first map string as ***lId, sText, lVisLev, fX, fY, fWidth, fAngle, lSymbolId*** and ***lColourId***. Data are retrieved in ascending order of ***lId***. The function **HyGetNext** can be used with this type of data to get the remaining map strings.

When the **HYUPDATE** flag is used ***sText, lVisLev, fX, fY, fWidth, fAngle, lSymbolId*** and ***lColourId*** are changed for the map string id specified in ***lId***.

When the **HYADD** flag is used a new map string is added to the database. ***lId, sText, lVisLev, fX, fY, fWidth, fAngle, lSymbolId*** and ***lColourId*** are all required for this type of function call.

When the **HYDELETE** flag is used the map string is removed from the database. ***lId*** must be specified for this call.

Example

```
BOOL      bOK;

long      lId, lVisLev, lSymbolId, lColourId;

double    fX, fY, fWidth;

HYHAND    hConnect;


/* Add a new map string */

lId = 43L;
lVisLev = 100L;
lSymbolId = -1L;
lColourId = 2L;
fX = 1034.4;
fY = 5643.23;
fAngle = 0.0;
fHeight = 20.5;

bOK = HyMapStrings ( hConnect, &lId, "Wallingford", &lVisLev, &fX, &fY, &fWidth,
&fAngle, &lSymbolId, &lColourId, HYADD );

HyCommit ( hConnect );
```

HyNameExists

```
BOOL HyNameExists ( hConnect, lId, lpszName, lpszTable, lpszIDColumn,
                    lpszNameColumn, lCount )

HYHAND   hConnect        /* HYDATA connection handle */
LPLONG   lId             /* Exclude id from check */
LPSTR    lpszName        /* Name to be checked for uniqueness*/
LPSTR    lpszTable       /* Table name */
LPSTR    lpszIDColumn    /* Name of the column holding the exclude id */
LPSTR    lpszNameColumn  /* Name of the column holding the names */
LPLONG   lCount          /* Number of times the lpszName has been found */
```

The **HyNameExists** function finds the number of times that an identifying name is used in a database table excluding the name in use for a single specified id. The function is used to check the uniqueness of a name in a table (for example to make sure that two stations do not have the same name). The check is case sensitive.

Parameter	Description
hConnect	HYDATA connection handle.
lId	The id of the item currently being edited so that the name associated with it can be excluded from the check
lpszName	The name to be checked for uniqueness. I.e. the new name the user wishes the table item to be called
lpszTable	The name of the database table to check.
lpszIDColumn	Name of the table's column which contains the **lId** to exclude.
lpszNameColumn	Name of the column holding the names to be checked against for uniqueness.
lCount	The number of times that ***lpszName*** has been found in column ***lpszNameColumn*** of table ***lpszTable*** excluding any instances of the id ***lId*** in column ***lpszIDColumn*** of the same table.

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 69

Comments

This case insensitive search is designed to help application programs to ensure that names remain unique within a table. If the function call was successful and **lCount** returned as zero, uniqueness is guaranteed. Obviously any check must exclude the current id since it will be possible for a user to edit a name back to its original value. In this case the name will be in the table but is obviously valid.

If successful this name check should be immediately followed by an update or insert to ensure the changes are available to other users.

Example

```
long      lId, lCount;

HYHAND    hConnect;

lId = 31;

if ( HyNameExists ( hConnect, &lId, "New station", "STATION", "ID", "NAME",
                    &lCount ) )
  {
  if ( lCount )
    {
    /* Disallow name as already in use */

    {
  else
    {
    /* Update table with new name */

    }
  }
else
  {
  /* Handle error */

  }
```

HyNextId

BOOL HyNextId (*hConnect, lpszTableName, lNextId, bPositive*)

```
HYHAND   hConnect        /* HYDATA connection handle */
LPSTR    lpszTableName   /* Table name */
LPLONG   lNextId         /* Next id */
BOOL     bPositive       /* Positive or negative id */
```

The **HyNextId** function returns the number the next free id to be used to insert a new entry into a table.

Parameter	Description
hConnect	HYDATA connection handle.
lpszTableName	A pointer to a null terminated string containing the name of the database table.
lNextId	Returned as the next free id for the table specified.
bPositive	Set to TRUE if the next positive id is required or FALSE if the next negative id is required

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 32

Comments

Positive ids are reserved for system use. Negative ids are for the use of the user.

The next free id is one greater than the value of the maximum currently in use in a table for a positive id or one less than the current minimum for a negative id. Positive ids start from a base of 1 while negative ids start from a base of -1. An id of zero is not used.

Gaps in the series are possible as the result of deletion. For example the id series, -5, -3, -2, -1, 1, 3, 4 is valid.

Example

```
long     lNextId;

HYHAND   hConnect;


iRet =      HyNextId ( hConnect, (LPSTR) "STATION", &lNextId, FALSE );
```

HyObjectAtts

```
BOOL HyObjectAtts ( hConnect, lObjectTypeId, lAttId, lpszAttName,
                    iAttType, iFlag )

HYHAND    hConnect          /* HYDATA connection handle */
LPLONG    lObjectTypeId     /* Object type id */
LPLONG    lAttId            /* Attribute id */
LPSTR     lpszAttName       /* Attribute name */
LPINT     iAttType          /* Attribute type */
int       iFlag             /* Function control flag */
```

The **HyObjectAtts** function controls information concerning object type attributes. Positive attribute ids are system defined values, negative attribute ids are user defined.

Parameter	Description
hConnect	HYDATA connection handle.
lObjectTypeId	Object type id
lAttId	Attribute id
lpszAttName	Attribute name
iAttType	Attribute type. Returned as one of the following six values: 1 - Character (max 254 characters) 2 - Long character (unlimited length string) 3 - Integer (long) 4 - Float (double) 5 - Date 6 - Picture
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first object type attribute **HYGETNEXT** - Get next object type attribute **HYADD** - Add a new object type attribute **HYDELETE** - Remove an object type attribute

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 26

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function returns details of the first object type attribute in as ***lAttId***, ***lpszName*** and ***iAttType*** for a given ***iObjectTypeId***. Data are retrieved in ascending order of ***lAttId***. Since the retrieval is complex the HyGetNext function cannot be used with this type of data. Use the **HYGETNEXT** flag on **HyObjectAtts** to get all remaining data.

When the **HYADD** flag is used a new attribute/object type relationship is added to the database. ***lObjectTypeId*** and ***lAttId*** are all required for this type of function call. Applications MUST ensure that only negative attributes are added to positive object type ids.

When the **HYDELETE** flag is used the attribute/object type relationship is removed from the database. Both ***lObjectTypeId*** and ***lAttId*** must be specified for this call.

Example

```
BOOL      bOK;

int          iAttType;

long      lObjectTypeId, lAttId;

char         sTmp [ 81 ];

HYHAND    hConnect;


/* Add a new attribute/object type relationship */

lObjectTypeId = 1
lAttId = -3L;
iType = 2L;

bOK = HyObjectAtts ( hConnect, &lObjectTypeId, &lAttId, sTmp, &iAttType, HYADD
);

HyCommit ( hConnect );
```

HyObjectTypes

BOOL HyObjectTypes (*hConnect, iObjectTypeId, lpszName*, iFlag)

```
HYHAND   hConnect          /* HYDATA connection handle */
LPINT    iObjectTypeId     /* Object type id */
LPSTR    lpszName          /* Object type name */
int      iFlag             /* Function control flag */
```

The **HyObjectTypes** function manages the table of HYDATA object types. Positive object type id's are reserved for system defined object types, negative id's for user defined object types.

Parameter	Description
hConnect	HYDATA connection handle.
iObjectTypeId	Object type id (+ve system defined object types, -ve for user defined object types)
lpszName	Name of the object type.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first object type **HYGETNEXT** - Get next object type

Returns

This function returns TRUE if successful or FALSE if the request fails.

Export ordinal

DLL export ordinal: 24

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and return details of the first object type as ***iObjectTypeId*** and ***lpszName***. The remaining units are retrieved by using **HyObjectTypes** with the **HYGETNEXT** flag. The end of the units is signified by a return value of FALSE.

Note that due to the complex nature of this retrieval, **HyGetNext** cannot be used in conjunction with this function; the **HYGETNEXT** flag must be used with **HyObjectTypes** to obtain a list of units. Data are retrieved in ascending order of ***iObjectTypeId***.

The current version of this library does not support the addition of user defined objects (ie those with a -ve object type id).

Example

```
BOOL     bOK;

int      iObjectTypeId;
char     sName [ 81 ];
char     sTmp [ 386 ];

HYHAND   hConnect;
```

```
bOK =       HyObjectTypes ( hConnect, &iObjectTypeId, sName, HYGETINIT );
sprintf ( sTmp, "Id %d Name %s", iObjectTypeId, sName );

while ( bOK )
  {
  bOK =     HyObjectTypes ( hConnect, &iObjectTypeId, sName, HYGETNEXT );
  sprintf ( sTmp, "Id %d Name %s", iObjectTypeId, sName );
  }

HyCommit ( hConnect );
```

HyPicture

```
BOOL HyPicture ( hConnect, lPicId, lFormat, lWidth, lHeight, lSizeMax,
                 lCompMeth, lColours, lPicture, lBuffSize, lBytesRead,
                 iFlag )

HYHAND    hConnect               /* HYDATA connection handle */
LPLONG    lPicId                 /* Picture id */
LPLONG    lFormat                /* Picture format */
LPLONG    lWidth                 /* Picture width */
LPLONG    lHeight                /* Picture height */
LPLONG    lSizeMax               /* Size before compression */
LPLONG    lCompMeth              /* Compression method */
LPLONG    lColours               /* Number of colours */
LPVOID    lPicture               /* Pointer to picture data */
size_t    lBuffSize              /* Size of data buffer */
unsigned long far * lBytes       /* Number of bytes */
int       iFlag                  /* Function control flag */
```

The **HyPicture** function controls the storage of picture data.

Parameter	Description
hConnect	HYDATA connection handle.
lPicId	Picture id
lFormat	Format: 1 = Bitmap (.BMP file)
lWidth	Width of picture in units applicable to format (eg pixels for bitmap)
lHeight	Height of picture in units applicable to format (eg pixels for bitmap)
lSizeMax	The size of the uncompressed image. If ***lSizeMax*** = ***lBytes*** when ***iFlag*** = **HYGETLONGPREP**, no compreesion has been used
lCompMeth	Compression method used to store the picture 1 =
lColours	Number of colours used in the picture if relevant to the format ***lFormat***
lPicture	Pointer to the picture data to be transfered
lBuffSize	Size of the buffer pointed to by ***lData*** (cannot be greater than 32,767)
lBytes	If ***iFlag*** = **HYGETLONGPREP**, ***lBytes*** is the total Picture size in bytes (compressed). If ***iFlag*** = **HYGETLONGREAD**, ***lBytes*** is the number of bytes read on the call and placed in the buffer. When a read is complete ***lBytes*** will be returned with a value of zero

iFlag	Function control flag which can take one of the following constant: **HYGETLONGPREP** - Prepare for a read **HYGETLONGREAD** - Read a block **HYGETLONGEND** - Terminate read **HYUPDATELONGPREP** - Prepare update **HYUPDATELONGADD** - Add an updated block **HYUPDATELONGEND** - End update **HYADDLONGPREP** - Prepare addition of new picture **HYADDLONGADD** - Add an updated block **HYADDLONGEND** - End addition of new picture **HYDELETE** - Remove a picture

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 33

Comments

Manipulation of picture is more complicated than single value data and is similar to that for handling lon character data as described for the function **HyAttLongChar**.

When adding a new picture or updating an existing picture, the picture data is supplied to the function in any number of blocks until all the whole picture is transfered to the database. The size of the block is determined by the parameter ***lBuffSize*** but cannot be greater than 32,767 bytes. The addition or update is a three stage process; preparation, adding blocks and termination. Other database calls must not be made before the operation is terminated by a call to **HyPicture** with a **HYADDLONGEND** or a **HYUPDATELONGEND** flag. The operation must still be terminated, even if either of the two earlier stages resulted in an error.

When reading a picture, the data are returned from the database in a series of blocks until all data are transfered. The size of the block is determined by the parameter ***lBuffSize*** but cannot be greater than 32,767 bytes. The read is a three stage process; preparation, reading blocks and termination. Other database calls must not be made before the operation is terminated by a call to **HyPicture** with a **HYGETLONGEND** flag. The operation must still be terminated, even if either of the two earlier stages resulted in an error.

Getting a picture

First use the **HYGETLONGPREP** flag to prepare for the read for a specified ***lPicId***. Note that ***lBytes*** is returned from the call and gives the total size of the character string that will be returned. ***lFormat, lWidth, lHeight, lSizeMax, lCompMeth and lColours*** are also returned on this type of acll.

Secondly use the **HYGETLONGREAD** flag in a loop to read the data. ***lPicture*** and ***lBuffSize*** must be supplied. ***lBytes*** is returned from each call and gives the total number of bytes returned in the buffer. When ***lBuffSize*** is zero the read is complete.

Finaly use the **HYGETLONGEND** flag to terminate the read. No other database call must made for this connect handle until the read is terminated. The read must still be terminated, even if a failure occurs.

Update existing picture/ add new picture

First use the **HYUPDATELONGPREP** or **HYADDLONGPREP** flag to prepare for the write for a specified ***lPicId***. "Update" should be used where the picture already exists on the database, "add" should be used where the particular ***lPicId*** does not exist. ***lFormat, lWidth, lHeight, lSizeMax, lCompMeth and lColours*** must be supplied.

Secondly use the **HYUPDATELONGADD** or **HYADDLONGADD** flag in a loop to write the data. ***lPicture*** and ***lBuffSize*** must be supplied.

Finaly use the **HYUPDATELONGEND** or **HYADDLONGEND** flag to terminate the write. No other database call must made for this connect handle until the write is terminated. The write must still be terminated, even if a failure occurs.

Deleting data

When the **HYDELETE** flag is used the entire reference to the picture is removed from the database for the specified values of ***lPicId***.

Example

```
BOOL      bOK;

long      lPicId, lFormat, lWidth, lHeight, lSizeMax, lCompMeth;
long      lColours;
unsigned long lSize;

char      sNewPic [ 10000 ];

HYHAND    hConnect;


/* Update an existing picture with data already stored in sNewPic */

lPicId = 1L;
lFormat = 1L;
lWidth = 100L;
lHeight = 100L;
lSizeMax = sizeof ( sNewPic );
lCompMeth = 0L;
lColours = 1L;


/* Prepare */

bOK = HyPicture ( hConnect, &lPicId, &lFormat, &lWidth, &lHeight, &lSizeMax,
&lCompMeth, &lColours, sNewPic, sizeof ( sNewPic ), &lSize, HYUPDATELONGPREP );

/* Add first block */

bOK = HyPicture ( hConnect, &lPicId, &lFormat, &lWidth, &lHeight, &lSizeMax,
&lCompMeth, &lColours, sNewPic, 5000, &lSize, HYUPDATELONGADD );

/* Add second block */
```

```
bOK = HyPicture ( hConnect, &lPicId, &lFormat, &lWidth, &lHeight, &lSizeMax,
&lCompMeth, &lColours, sNewPic [ 5000 ], 5000, &lSize, HYUPDATELONGADD );

/* Terminate */

bOK = HyPicture ( hConnect, &lPicId, &lFormat, &lWidth, &lHeight, &lSizeMax,
&lCompMeth, &lColours, sNewPic, sizeof ( sNewPic ), &lSize, HYUPDATELONGEND );


HyCommit ( hConnect );
```

HyRatData

BOOL HyRatData (*hConnect, lTSId, lRatId, ld, fValue, iFlag*)

```
HYHAND      hConnect           /* HYDATA connection handle */
LPLONG      lTSId              /* Time series id */
LPLONG      lRatId             /* Rating equation id */
LPLONG      lId                /* Parameter id */
double far* fValue             /* Parameter value */
int         iFlag              /* Function control flag */
```

The **HyRatData** function controls information concerning rating equation parameters.

Parameter	Description
hConnect	HYDATA connection handle.
lRatId	Rating equation id
lId	Parameter id: h is stage: Power rating: 1 = a_1 $q = a . (h + c) b)$ 2 = b_1 3 = c_1 4 = h_{max} 5 = a_2 etc Polynomial rating: 1 = hMin 2 = hMax 3 = a_0 $q = a_0 + a_1.h + a_1.h^2 + a_1.h^3$ 4 = a_1 5 = a_2 6 = a_3 etc Rating table: (not implemented in version 4.0) 1 = hmin 2 = hmax 3 = h_1 4 = q_1 5 = h_2 6 = q_2 etc
fValue	Parameter value
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first rating **HYGETNEXT** - Get next rating **HYUPDATE** - Update rating **HYADD** - Add a new rating **HYDELETE** - Remove a rating **HYDELETEBLOCK** - Remove all ratings for a time series

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 56

Comments

Use the **HYGETINIT** flag on the first call to prepare the query for a specified ***lTSId*** and ***lRatId*** and the function return details of the first parameter as ***lId***, and ***fValue***. Data are retrieved in ascending order of ***lId***. Use the **HYGETNEXT** flag to get all remaining data.

When the **HYUPDATE** flag is used ***fValue*** is changed for the specified ***lTSId***, ***lRatId*** and ***lId***.

When the **HYADD** flag is used a new parameter is added to the database. All parameters are required for this type of function call.

When the **HYDELETE** flag is used the parameter identified by ***lTSId***, ***lRatId*** and ***lId*** is removed from the database.

The **HYDELETEBLOCK** flag removes all rating parameters from the database identified by ***lTSId***.

Example

```
BOOL      bOK;

long      lId, lRatId, lTSId;

double    fValue;

HYHAND    hConnect;


/* Add a new parameter */

lId = 23L;
lRatId = 12L;
lTSId = 15L;

fValue = 10.34;

bOK = HyRatData ( hConnect, &lTSId, &lRatId, &lId, &fValue, HYADD );

HyCommit ( hConnect );
```

HyRatDef

```
BOOL HyRatDef ( hConnect, lId, lpszName, lTSId, lRatTypeId, fSDate,
                fEDate, lSDay, lEDay, lpszComments, iFlag )

HYHAND     hConnect         /* HYDATA connection handle */
LPLONG     lId              /* Rating id */
LPSTR      lpszName         /* Rating name */
LPLONG     lTSId            /* Time series id */
LPLONG     lRatTypeId       /* Rating type id */
double far *  fSDate        /* Start date (and time) of rating */
double far *  fEDate        /* End date (and time) of rating */
LPLONG     lSDay            /* Start day number in year */
LPLONG     lEDay            /* End day number in year */
LPSTR      lpszComments     /* Comments */
int        iFlag            /* Function control flag */
```

The **HyRatDef** function controls information concerning the definition of rating equations.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Rating id (unique across all time series ids)
lpszName	Rating name
lTSId	Time series id

lRatTypeId	Rating type id:
	HYRATTYPEPOWER = Power rating($q = a . (h + b)^{c}$)
	HYRATTYPEPOLY = Polynomial

fSDate	Date the rating equation becomes valid
fEDate	End of period that the rating applies
lSDay	Start day within year for a seasonal rating. Non seasonal set to 1
lEDay	End day within year for a seasonal rating. Non seasonal set to 366. Note ***lSDay*** can be more than ***lEDay*** for seasons over the year end
lpszComments	Comments on the rating
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first rating **HYGETNEXT** - Get next rating **HYUPDATE** - Update rating **HYADD** - Add a new rating **HYDELETE** - Remove a rating

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 56

Comments

Use the **HYGETINIT** flag on the first call to prepare the query for a specified ***lTSId*** and the function return details of the first gauging as ***lId, lpszName,***

lRatTypeId, fSDate, fEDate, lSDay, lEDay and ***lpszComments***. Data are retrieved in ascending order of ***fSDate***. Use the **HYGETNEXT** flag to get all remaining data.

When the **HYUPDATE** flag is used ***lpszName, fSDate, fEDate, lSDay, lEDay*** and ***lpszComments*** are changed for the specified ***lId***.

When the **HYADD** flag is used a new rating definition is added to the database. All parameters are both for this type of function call.

When the **HYDELETE** flag is used the rating identified by ***lId*** is removed from the database. The rating definition parameters are also deleted.

When the **HYDELETEBLOCK** flag is used all ratings for ***lTSId*** is removed from the database. The rating definition parameters are also deleted for all these ratings.

Example

```
BOOL      bOK;

long      lId, lRatTypeId, lTSId, lSDay, lEDay;

double    fSDate, fEDate;

char      sName [ 20 ];
char      sComments [ 257 ];

HYHAND    hConnect;


/* Add a new rating definition */

lId = 23L;
lTSId = 15L;
lRatTypeId = HYRATTYPEPOWER;
lSDay = 1L;
lEDay = 366L;

fSDate = 34355.5;
fEDate = 36234.5;

lstrcpy ( sComment, "New rating" );
lstrcpy ( sName, "B" );

bOK = HyRatDef ( hConnect, &lId, sName, &lTSId, &lRatTypeId, &fSDate, &fEDate,
&lSDay, &lEDay, sComments, HYADD );

HyCommit ( hConnect );
```

HyRiverLocs

BOOL HyRiverLocs (***hConnect, lId, lDsId, lUsId, fX, fY, fElev, fChain, iFlag***)

```
HYHAND     hConnect          /* HYDATA connection handle */
LPLONG     lId               /* River id */
LPLONG     lDsId             /* Downstream location id */
LPLONG     lUsId             /* Upstream location id (main channel) */
float far *    fX            /* X co-ordinate */
float far *    fY            /* Y co-ordinate */
float far *    fElev         /* Elevation */
float far *    fChain        /* Chainage to downstream location */
int        iFlag             /* Function control flag */
```

The **HyRiverLocs** function controls information concerning river locations.

Parameter	Description
hConnect	HYDATA connection handle.
lId	River id.
lDsId	Downstream location id (-1 if at end of river)
lUsId	Upstream location id of the main channel. If there is no upstream location this ***lUsId*** is zero
fX	X co-ordinate for plotting on map
fY	Y co-ordinate for plotting on map
fElev	Elevation of the location. ***fElev*** is in internal HYDATA units of metres
fChain	Chainage between this location and the location downstream (zero if at end of river). ***fChain*** is in internal HYDATA units of metres.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first river **HYGETNEXT** - Get next river **HYUPDATE** - Update river name **HYADD** - Add a new river **HYDELETE** - Remove a station

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL-export ordinal: 38

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first river location as ***lId, lDsId, lUsId, fX, fY, fElev*** and ***fChain***. Data are retrieved in ascending order of ***lId***. The function **HyGetNext** can be used with this type of data to get the remaining rivers locations.

When the **HYUPDATE** flag is used ***lDsId, lUsId, fX, fY, fElev*** and ***fChain*** are changed for the river location id specified in ***lId***.

When the **HYADD** flag is used a new river location is added to the database. ***lId, lDsId, lUsId, fX, fY, fElev*** and ***fChain*** are all required for this type of function call.

When the **HYDELETE** flag is used the location is to be removed from the database. ***lId*** must be specified for this call.

Example

```
BOOL      bOK;

long      lId, lUsId, lDsId;

double    fX, fY, fElev, fChain;

HYHAND    hConnect;


/* Add a new river location */

lId = 104L;
lDsId = 103L;
lUsId = 105L;
fX = 1234.5;
fY = 223311.2;
fElev = 102.2;
fChain = 23456.3;

bOK = HyRiverLocs ( hConnect, &lId, &lDsId, &lUsId, &fX, &fY, &fElev, &fChain,
HYADD );

HyCommit ( hConnect );
```

HyRivers

BOOL HyRivers (***hConnect, lId, lpszName, lLocId, iFlag***)

```
HYHAND   hConnect        /* HYDATA connection handle */
LPLONG   lId             /* River id */
LPSTR    lpszName        /* River name */
LPLONG   lLocId          /* River location id */
int      iFlag           /* Function control flag */
```

The **HyRivers** function controls information concerning river definition.

Parameter	Description
hConnect	HYDATA connection handle.
lId	River id.
lpszName	River name
lLocId	River location id. This is the most downstream location id of the river. All river locations upstream of this id belong to this river system.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first river **HYGETNEXT** - Get next river **HYUPDATE** - Update river name **HYADD** - Add a new river **HYDELETE** - Remove a station

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 37

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first river as ***lId, lpszName*** and ***lLocId***. Data are retrieved in ascending order of ***lId***. The function **HyGetNext** can be used with this type of data to get the remaining rivers.

When the **HYUPDATE** flag is used ***lpszName*** and ***lLocId*** are changed for the river id specified in ***lId***.

When the **HYADD** flag is used a new river is added to the database. ***lId, lpszName***, and ***lLocId*** are all required for this type of function call.

When the **HYDELETE** flag is used the river is to be removed from the database. ***lId*** must be specified for this call. Any other data associated with the river is not deleted; it is the responsibility of the application to make sure that it is.

Example

```
BOOL      bOK;
```

```
long      lId, lLocId;

HYHAND    hConnect;


/* Add a new river */

lId = 4L;
lLocId = 103L;

bOK = HyRivers ( hConnect, &lId, "New river", &lLocId, HYADD );

HyCommit ( hConnect );
```

HyRollback

BOOL HyRollback (*hConnect*)

```
HYHAND    hConnect          /* HYDATA connection handle */
```

The **HyRollback** function rollsback all transactions outstanding for the application and allows other users access to data modified data.

Parameter	Description
hConnect	HYDATA connection handle associated with this connection.

Returns

The function returns TRUE if successful or FALSE if an error occured.

Export ordinal

DLL export ordinal: 21

Comments

This function or **HyCommit** should be used as soon as possible after data are abstracted or altered to enable access by other users.

Example

```
BOOL      bOK;
HYHAND    hConnect;

if ( HyUpdateUnits ( hConnect, 4L, "New name", 25.4, 0.2, 3 ) )
  HyCommit ( hConnect );
else
  HyRollback ( hConnect );
```

HySelObj

```
BOOL HySelObj ( hConnect, hWnd, lpszDB, lObjectTypeId, lObjectId,
                lObjectSubTypeId, lpszName, lTotal, bChanged, lIndex,
                iFlag )

HYHAND    hConnect           /* HYDATA connection handle */
HWND      hWnd               /* Window handle of selected list */
LPSTR     lpszDB             /* Database name */
LPLONG    lObjectTypeId      /* Object type id */
LPLONG    lObjectId          /* Object id */
LPLONG    lObjectSubTypeId   /* Object sub type id */
LPSTR     lpszName           /* Object name */
LPLONG    lTotal             /* Total number of selected objects */
BOOL far *     bChanged      /* TRUE if object list has changed */
LONG      lIndex             /* Sequence number of object */
int       iFlag              /* Function control flag */
```

The **HySelObj** function controls the selected object list.

Parameter	Description
hConnect	HYDATA connection handle.
hWnd	Handle of the window which displays the list of selected objects. Only required with ***iFlag*** = HYSELINIT..
lpszDB	Name of the database that the object has been selected from
lObjectTypeId	Object type id
lObjectId	Object id
lObjectSubTypeId	Object sub type id.
lpszName	Name of object
lTotal	Number of currently selected objects
bChanged	TRUE if the selected object list has changed since the current ***hConnect*** last called HySelObj with the flag HYSELINQ.
lIndex	Sequence number of the object on the selected list of selected objects
iFlag	Function control flag which can take one of the following constant: **HYSELINIT** - Initialise the function **HYSELADD** - Add an object to the list **HYSELREM** - Remove an object from the list **HYSELCLEAR** - Clear the selected object list **HYSELINQ** - Inquire details of selected object **HYSELEND** - End object selection facilities

Returns

Returns TRUE if the call was successful, FALSE if not. A return of FALSE when ***iFlag*** = **HYSELINQ** indicates the requested object was not on the list.

Export ordinal

DLL export ordinal: 46

Comments

The **HYSELINIT** flag is used to initialise the object selection facilities. Parameters required are ***hConnect*** and ***hWnd***. ***hWnd*** is the handle of the window that displays the list of selected objects. Only the HYDATA program manager should call the function with this flag[*] .

An object is added to the selected list with the **HYSELADD** flag. Parameters required are ***hConnect, lpszDB, lObjectTypeId, lObjectId, lObjectSubTypeId*** and ***lpszName***. ***lTotal*** is returned as the new total number of objects and ***bChanged*** is returned as TRUE. Since different connects can be connected to different databases, the name of the database that the object has been selected from is returned. If the object is displayed on the map its symbol will automatically change colour to indicate that it has been selected.

An object is removed from the selected list with the **HYSELREM** flag. Parameters required are ***hConnect, lpszDB, lObjectTypeId*** and ***lObjectId***. ***lTotal*** is returned as the new total number of objects and ***bChanged*** is returned as TRUE. It is not necessary to specify ***lObjectSubTypeId*** since ***lObjectId*** is unique for each lObjectTypeId regardless of the subtype. If the object is displayed on the map its symbol will automatically change colour to indicate that it is no longer selected.

The selected object list is cleared of all entries when the **HYSELCLEAR** flag is used. ***lTotal*** is returned as the new total number of objects (zero) and ***bChanged*** is returned as TRUE. All map symbols affected are re-drawn in the 'not selected' colour.

The **HYSELINQ** flag is used to inquire details of a selected object. Parameters required are ***hConnect*** and ***lIndex***. Parameters returned are ***bChanged, lTotal, lObjectTypeId, lObjectId, lObjectSubTypeId, lpszDB*** and ***lpszName***. If the sequence number of the object given by ***lIndex*** is not valid the function returns FALSE and ***lTotal*** is the only parameter returned. The value of ***bChanged*** must be checked after each call in case the selected object list has changed. If the list has changed ***bChanged*** is returned as TRUE for this call only; subsequent calls with the **HYSELINQ** flag will return ***bChanged*** as FALSE.

The **HYSELEND** flag is used to terminate the object selection facilities. The only parameters required is ***hConnect***. Only the HYDATA program manager should call the function with this flag.

Example

```
BOOL      bOK, bChanged;

char         sDB [ 12 ], sName [ 81 ];

long      lObjectTypeId, lObjectId, lObjectSubTypeId, lTotal;

HYHAND    hConnect;


/* Inquire the first selected object */

bChanged = TRUE;

while ( bChanged )
```

[*] It appears that despite the above comment, that the function must be called with the HYSELINIT parameter in order to allocate memory used by this function

```
  {
  bOK  =  HySelObj  (  hConnect,  (HWND)  0,  sDB,  lObjectTypeId,  lObjectId,
  lObjectSubTypeId, sName, lTotal, bChanged, 1L, HYSELINQ );
  }

if ( bOK )
  {
  /* Do whatever with sDB, lObjectTypeId, lObjectId, lObjectSubTypeId, sName and
  lTotal */

  }
else
  {
  /* Only returned parameter is lTotal */

  }
```

HySpot

```
BOOL HySpot ( hConnect, lId, lTSId, fReadTime, fFlow, fPer,
                lpszComments, iFlag )

HYHAND     hConnect           /* HYDATA connection handle */
LPLONG     lId                /* Spot gauging id */
LPLONG     lTSId              /* Time series id */
double far * fReadTime        /* Date and time of spot gauging */
double far * fFlow            /* Total discharge */
double far * fPer             /* Percentile */
LPSTR      lpszComments       /* Comments */
int        iFlag              /* Function control flag */
```

The **HySpot** function controls information concerning river spot gaugings.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Spot gauging id (unique across all time series ids)
lTSId	Time series id
fReadTime	Date and time the spot gauging was undertaken
fFlow	Total discharge of the spot gauging
fPer	Percentile associated with the flow
lpszComments	Comments on the spot gauging
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first spot gauging **HYGETNEXT** - Get next spot gauging **HYUPDATE** - Update spot gauging **HYADD** - Add a new spot gauging **HYDELETE** - Remove a spot gauging

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 54

Comments

Use the **HYGETINIT** flag on the first call to prepare the query for a specified ***lTSId*** and the function return details of the first spot gauging as ***lId***, ***fReadTime***, ***fFlow***, ***fPer*** and ***flpszComments***. Data are retrieved in ascending order of ***fReadTime***. Use the **HYGETNEXT** flag to get all remaining data.

When the **HYUPDATE** flag is used ***fReadTime***, ***fFlow***, ***fPer*** and ***lpszComments*** are changed for the specified ***lId***.

When the **HYADD** flag is used a new spot gauging is added to the database. All parameters are both for this type of function call.

When the **HYDELETE** flag is used the spot gauging identified by ***lId*** is removed from the database.

Example

```
BOOL      bOK;

long      lId, lTSId;

double    fReadTime, fFlow, fPer;

char      sComments [ 257 ];

HYHAND    hConnect;


/* Add a new spot gauging */

lId = 23L;
lRatId = 0L;

fFlow = 20.34;
fPer = 10.8;

fReadTime = 34356.5;

lstrcpy ( sComment, "New spot gauging" );

bOK = HySpot (hConnect, &lId, &lTSId, &fFlow, &fPer, sComments, HYADD);

HyCommit ( hConnect );
```

HyStations

```
BOOL HyStations ( hConnect, lId, lpszNumA, lpszNumB, lpszName, lTypeId,
                  fX, fY, fAngle, lRiverLocId, iFlag )

HYHAND      hConnect        /* HYDATA connection handle */
LPLONG      lId             /* Station id */
LPSTR       lpszNumA        /* Station number (primary) */
LPSTR       lpszNumB        /* Station number (secondary) */
LPSTR       lpszName        /* Station name */
LPLONG      lTypeId         /* Station type id */
double far * fX             /* Map x co-ordinate */
double far * fY             /* Map y co-ordinate */
double far * fAngle         /* Map name drawn angle */
LPLONG      lRiverLocId     /* River location id */
int         iFlag           /* Function control flag */
```

The **HyStations** function controls information concerning station definition.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Station id. This must be a unique id regardless of station type (for example station id 2 cannot exist for both station type 3 and station type 4)
lpszNumA	Primary station number according to the user's prefered numbering system
lpszNumB	Secondary station number according to the user's prefered numbering system
lpszName	Station name
lTypeId	Station type id
fX	Map x co-ordinate
fY	Map y co-ordinate
fAngle	Angle at which the station name will be draw on the map
lRiverLocId	River location id. If the station is located on the river network (eg gauging station, abstraction) this is set to the river location id where the station is situated. The station is plotted on the map at co-ordinates ***fX*** and ***fY*** for the station rather than the river location co-ordinates. ***lRiverLocId*** is zero if the station is not located on the river.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first station **HYGETNEXT** - Get next station **HYUPDATE** - Update station name **HYADD** - Add a new station **HYDELETE** - Remove a station

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 36

Comments

The integer attibute HYATTNUMSYS, which defines the preferred station numbering system for each user, is maintained by the program manager. Function **HyStations** uses the value of this attribute at connect time to determine which station numbering system is primary and which is secondary for the user. This function uses this attribute to determine which station number is allocated to ***lpszNumA*** and ***lpszNumB***. If the user changes this preference, ***lpszNumA*** and ***lpszNumB*** will be reversed on the next connect.

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first station as ***lId***, ***lpszNumA***, ***lpszNumB***, ***lpszName***, ***fX***, ***fY***, ***fAngle***, ***lRiverLocId*** and ***lTypeId***. Data are retrieved in ascending order of ***lTypeId*** and then for ***lId*** within each ***lTypeId***. The function **HyGetNext** can be used with this type of data to get remaining stations.

When the **HYUPDATE** flag is used ***lpszNumA***, ***lpszNumB***, ***lpszName***, ***fX***, ***fY***, ***fAngle***, and ***lRiverLocId*** are changed for the station id specified in ***lId***.

When the **HYADD** flag is used a new station is added to the database. ***lId***, ***lpszNumA***, ***lpszNumB***, ***lpszName***, ***lTypeId***, ***fX***, ***fY***, ***fAngle***, ***lRiverLocId*** are all required for this type of function call. Note that ***lId*** must be unique; the same value of ***lId*** must not be used for different station types.

When the **HYDELETE** flag is used the station is to be removed from the database. ***lId*** must be specified for this call. Any other data associated with the station is not deleted; it is the responsibility of the application to make sure that it is.

Example

```
BOOL     bOK;

long     lId, lTypeId, lRiverLocId;

double   fX, fY, fAngle;

HYHAND   hConnect;


/* Add a new station */

lId = 234L;
lTypeId = 1L;
fX = 1254.3;
fY = 25.3;
fAngle = 0.0;
lRiverLocId = 5443L;

bOK = HyStations ( hConnect, &lId, "A123", "34215632", "New station", &lTypeId,
&fX, &fY, &fAngle, &lRiverLocId, HYADD );

HyCommit ( hConnect );
```

HyStationTypes

BOOL HyStationTypes (*hConnect, lTypeId, lpszName, iFlag*)

```
HYHAND   hConnect        /* HYDATA connection handle */
LPLONG   lTypeId         /* Station type id */
LPSTR    lpszName        /* Station type name */
int      iFlag           /* Function control flag */
```

The **HyStationTypes** function controls information concerning station types.

Parameter	Description
hConnect	HYDATA connection handle.
lTypeId	Station type id
lpszName	Station type name
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first station type **HYGETNEXT** - Get next station type **HYUPDATE** - Update station type name **HYADD** - Add a new station type **HYDELETE** - Remove a station type

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 35

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first station type as ***lTypeId***, and ***lpszName***. Data are retrieved in ascending order of ***lTypeId***. Since the retrieval is complex the **HyGetNext** function cannot be used with this type of data. Use the **HYGETNEXT** flag on **HyStationTypes** to get all remaining data.

When the **HYUPDATE** flag is used ***lpszName*** is changed for the station type id specified in ***lTypeId***. Applications must not change system defined station type names (ie ***lTypeId*** MUST be negative).

When the **HYADD** flag is used a new station type is added to the database. ***lTypeId***, and ***lpszName*** are both required for this type of function call. Applications must only add user defined station types (ie ***lTypeId*** MUST be negative).

When the **HYDELETE** flag is used the station type is removed from the database together with any stations of that type. ***lTypeId*** must be specified for this call. Any other data associated with stations of the type that has been removed is not deleted; it is the responsibility of the application to make sure that it is.

Example

```
BOOL      bOK;

long      lTypeId;

HYHAND    hConnect;

/* Add a new station type */

lTypeId = -3L;

bOK = HyStationType ( hConnect, &lTypeId, "New type", HYADD );

HyCommit ( hConnect );
```

HyStructCd

BOOL HyStructCd (*hConnect, lId, fXValue, fCdValue, iFlag*)

```
HYHAND        hConnect        /* HYDATA connection handle */
LPLONG        lId             /* Cd definition id */
double far *  fXValue         /* X value for Cd */
double far *  fCdValue        /* Value of Cd at fXValue */
int           iFlag           /* Function control flag */
```

The **HyStructCd** function controls information concerning the definition of Cd for hydraulic structures.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Cd definition id
fXValue	X value for which the corresponding ***fCdValue*** is valid
fCdValue	The value of Cd for the X value given in ***fXValue***
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get the first Cd definition point **HYGETNEXT** - Get next Cd definiton **HYUPDATE** - Update the Cd value **HYADD** - Add a new Cd definition point **HYDELETE** - Remove all Cd definitions of a given id

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 58

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first Cd definition point for a given Cd id as specified in ***lId***. ***fXValue*** and ***fCdValue*** are returned. Data are retrieved in ascending order of ***fXValue***. The **HyGetNext** function can be used with this type of data to get all remaining structures at the station.

When the **HYUPDATE** flag is used ***fCdValue*** is changed for a given ***lId*** and ***fXValue***.

When the **HYADD** flag is used a new Cd definition point is added to the database. All parameters are required for this type of function call.

When the **HYDELETE** flag is used ALL Cd definition points are removed from the database for a given ***lId***. ***lId*** must be specified for this call.

Example .

```
BOOL      bOK;
```

```
long      lId;

double    fXValue, fCdValue;

HYHAND    hConnect;


/* Add a new Cd definition point */

lId = 14L;
fXValue = 1.2;
fCDValue = 0.67;

bOK = HyStructCd ( hConnect, &lId, &fXValue, &fCdValue, HYADD );

HyCommit ( hConnect );
```

HyStructCdType

BOOL HyStructCdType (*hConnect, lId, sName, sTypeFlag, lUnitId, fMin, fMax, iFlag*)

```
HYHAND      hConnect          /* HYDATA connection handle */
LPLONG      lId               /* Structure cd type id */
LPSTR       sName             /* Structure cd type name */
LPSTR       sTypeFlag         /* Set to 'Y' if multiple */
LPLONG      lUnitId           /* Unit id for display of cd values */
double far * fMin             /* Minimum allowable value for CD */
double far * fMax             /* Maximum allowable value for CD */
int         iFlag             /* Function control flag */
```

The **HyStructCdType** function controls information concerning the pre-loaded information on types of Cd for hydraulic structures.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Structure type id
sName	Name of hydraulic structure Cd type
sTypeFlag	Set to 'Y' if more than one calibration value
lUnitId	Unit id for the display of this type of Cd
fMin	Minimum value that this Cd type can take
fMax	Maximum value that this Cd type can take
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first Cd type **HYGETNEXT** - Get next Cd type

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 61

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first type of hyraulic structure Cd. ***lId, sName, sTypeFlag, lUnitId, fMin*** and ***fMax*** are returned. Data are retrieved in ascending order of ***lId***. The **HyGetNext** function **cannot** be used with this type of data because the query is complex; use the **HYGETNEXT** flag with the current function (**HyStructCdType**) to get remaining Cd types.

Note that this table is pre-loaded and cannot be edited by the user.

Example

```
BOOL     bOK;

int      iFlag;

long     lId, lUnitId;
```

```
char      sName [ 81 ];
char      sTypeFlag [ 2 ];

double    fMin, fMax;

HYHAND    hConnect;


/* Get hydraulic structure Cd types */

bOK = TRUE;
iFlag = HYGETINIT;

while ( bOK )
  {
  bOK = HyStructCdType (hConnect, &lId, sName, sTypeFlag, lUnitId, fMin, fMax,
          iFlag);
  iFlag = HYGETNEXT;
  if ( bOK )
    {
      ... do something with Cd id
    }
  }

HyCommit ( hConnect );
```

HyStructData

BOOL HyStructData (***hConnect, lStructId, lParamId, fValue, iFlag***)

```
HYHAND        hConnect         /* HYDATA connection handle */
LPLONG        lStructId        /* Structure id */
LPLONG        lParamId         /* Structure parameter id */
double far *  fValue           /* Value of parameter */
int           iFlag            /* Function control flag */
```

The **HyStructData** function controls information concerning the parameters for hydraulic structures.

Parameter	Description
hConnect	HYDATA connection handle.
lStructId	Structure id
lParamId	Structure parameter id
fValue	The paremeter value
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first parameter value for structure **HYGETNEXT** - Get next parameter **HYUPDATE** - Update the parameter value **HYADD** - Add a new parameter **HYDELETE** - Remove all parameters for a structure

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 59

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first parameter for a structure as specified in ***lStructId***. ***lParamId*** and ***fValue*** are returned. Data are retrieved in ascending order of ***lParamId***. The **HyGetNext** function can be used with this type of data to get all remaining structures at the station.

When the **HYUPDATE** flag is used ***fValue*** is changed for a given ***lStructId*** and ***lParamId***.

When the **HYADD** flag is used a new structure parameter is added to the database. All parameters are required for this type of function call.

When the **HYDELETE** flag is used ALL parameters are removed from the database for a given ***lStructId***. ***lStructId*** must be specified for this call.

Example

```
BOOL      bOK;
```

```
long      lStructId, lParamId;

double    fValue;

HYHAND    hConnect;


/* Add a new structure parameter */

lStructId = 4L;
lParamId = 1L;
fValue = 1.254;

bOK = HyStructData (hConnect, &lStructId, &lParamIde, &fValue, HYADD);

HyCommit ( hConnect );
```

HyStructFlow

```
int HyStructFlow (hConnect, lStationId, fUsLev, fDsLev, fGateLev,
                  fFlow, iFlag)

HYHAND    hConnect            /* HYDATA connection handle */
LPLONG    lStationId          /* Station id */
double    fUsLev              /* Upstream level */
double    fDsLev              /* Downstream level */
double far *  fGateLev        /* Gate level array */
double far *  fFlow           /* Value of flow returned */
int       iFlag               /* Function control flag */
```

The **HyStructFlow** function calculates the flow through all hydraulic structures at a gauging station and uses the standard INCA library INCA.DLL to undertake the conversion of water (and gate) level to flow.

Parameter	Description
hConnect	HYDATA connection handle.
lStationId	Station id
fUsLev	Upstream water level with the datum already added. This parameters is always required when calculating flow. Supply in internal units. Set to the constant **HYSTRUCTFLOWHUNAV** if this level is unavailable.
fDsLev	Downstream water level with the datum already added. This parameter is only required for some structures in certain conditions. Supply in internal units. Set to the constant **HYSTRUCTFLOWHUNAV** if this level is unavailable.
fGateLev	Gate level with the datum already added. This parameter is only required for gated structures. Supply in internal units. This is an array of gate levels, one for each structure and must be ordered according to the hydraulic structure's order number at the gauging station (see the **HyStructure** function). If the type of structure requires a gate (and this can be checked using the **HyStructType** function) this value must be supplied. If there is no value and one is required then the function which is about to call **HyStructFlow** must report an error and not proceed with the call. If no gate level is required set this parameter to **HYSTRUCTFLOWHUNAV**.
fFlow	The flow value returned in internal units.
iFlag	Function control flag which can take one of the following constant: **HYSTRUCTFLOWINIT** - Load structure parameters **HYSTRUCTFLOWCALC** - Calculate structure flow **HYSTRUCTFLOWDONE** - Free resources

Returns

Returns an integer status code. If zero there is no error and the flow is in fFlow. If non zero use the function HyStructError to get the error text in the current langauge.

Export ordinal

DLL export ordinal: 65

Comments

The **HYSTRUCTFLOWINIT** flag is used on the first call for a particular gauging station id to set up the parameters which are used to calculate the flow. The flag may be used if the gauging station is not changed but it is known that the hydraulic structure parameters have changed (eg in the Gauging and rating module). Only ***hConnect, lStationId*** and ***iFlag*** need to be supplied.

When the **HYSTRUCTFLOWCALC** flag is used to calculate the flow through the structures. ***lStationId, fUsLev, fDsLev*** and the gate setting array ***fGateLev*** must all be supplied. The downstream and gate levels may not be required (depending on structure type and degree of drowning). The datum must be added to these values. The calculated flow is returned as ***fFlow;*** if an error occurs this is signified by the function returning a non zero value. ***lStationId*** may be changed without calling the function with the **HYSTRUCTFLOWINIT** flag. Note that if structure flows are to be calculated at a number of gauging stations it is much more efficient keep ***lStationId*** constant between function calls and vary other parameters more often.

The **HYSTRUCTFLOWDONE** flag must be called after all flow calculations are complete to free resources.

Notes on the return status code:

1-100	error codes embedded in INCA.DLL
101-200	new structure-related errors e.g. no structures at a station
201-300	database errors when computing structure flow e.g. timeout
301-400	system errors when computing flow e.g. memory allocation

The error codes embedded in INCA.DLL are shifted left two places i.e. multiplied by 4, and the bottom bits set as follows:

00	information - only used when status code is 0 i.e. success
01	warning e.g. flow calculation may be inaccurate - an approximate flow is returned
10	error e.g. cannot calculate flow for given levels, but it should be possible to calculate flows for other levels for the same structure(s)
11	fatal e.g. cannot calculate flow because structure parameter value is invalid or missing - it is impossible to calculate the flows for this station under any circumstances

The idea was that computation proceeds on a warning, is terminated on an error but other values in the time series may be calculated, and is completely terminated on a fatal error i.e. there is no point in passing other time series levels to the function for this station. Accordingly, the new error codes above have also been shifted left 2 places and turned into warnings, errors, or fatal errors as appropriate. In fact, all the new errors are regarded as fatal.

Example

```
BOOL      bOK;

long      lStationId, lStatus, lStatus;

double    fUsLev, fDsLev, fGateLev [ 2 ], fFlow;

char      sStatus [ 81 ];

HYHAND    hConnect;


/* Calculate flow */

lStationId = 4L;
fUsLev = 1.23;
fDsLev = 1.254;
fGateLev [ 0 ] = 1.5;
fGateLev [ 1 ] = 1.3;

if ( lStatus = (long) HyStructFlow ( hConnect, lStationId, fUsLev, fDsLev,
fGateLev, &fFlow, HYSTRUCTFLOWCALC ) )
  {
    /* Handle error */
    HyStructError ( hConnect, lStatus, sStatus )
  }
```

HyStructError

```
void HyStructError (hConnect, lStructTypeId, lStatus, sString)

HYHAND    hConnect          /* HYDATA connection handle */
LPLONG    lStatus           /* Return status from a call to HyStructFlow */
LPSTR     sString           /* Returned string */
```

The **HyStructError** returns the string associated with a non zero status code after calling **HyStructFlow**.

Parameter	Description
hConnect	HYDATA connection handle.
lStatus	The non zero (error or warning) status value returned by function **HyStructFlow** (see description of function **HyStructFlow**). No bit shifting is required - use the return status cast as LONG.
sString	The text string describing the error or warning (in the current operating langauge). The string must be at least 255 bytes long to accomodate the the largest possible message.

Returns

Void function - no return value.

Export ordinal

DLL export ordinal: 68

Comments

See description of function **HyStructFlow** for status return code meanings.

Example

See of function **HyStructFlow**.

HyStructParam

```
BOOL HyStructParam (hConnect, lStructTypeId, lParamId, lPhraseId,
                    lUnitId, iFlag)

HYHAND   hConnect          /* HYDATA connection handle */
LPLONG   lStructTypeId     /* Structure type id */
LPLONG   lParamId          /* Structure parameter id */
LPLONG   lPhraseId         /* Structure phrase id */
int      iFlag             /* Function control flag */
```

The **HyStructParam** function controls information concerning the pre-loaded information relating to hydraulic structure parameters.

Parameter	Description
hConnect	HYDATA connection handle.
lStructTypeId	Structure type id
lParamId	Structure parameter id (sequence number of parameter for structure)
lPhraseId	Structure phrase id
lUnitId	Parameter unit id for display
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first parameter **HYGETNEXT** - Get next parameter

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 63

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first parameter for this type of hyraulic structure. ***lParamId, lPhraseId*** and ***lUnitId*** are returned for the ***lStructType*** supplied. Data are retrieved in ascending order of ***lParamId***. The **HyGetNext** function can be used with this type of data to get the remaining parameters.

Note that this table is pre-loaded and cannot be edited by the user.

Example

```
BOOL      bOK;

int       iFlag;

long      lStructTypeId, lParamId, lPhraseId, lUnitId;

HYHAND    hConnect;


/* Get hydraulic structure parameters */
```

```
bOK = TRUE;
iFlag = HYGETINIT;
lStructType = 2L;

while ( bOK )
  {
  bOK = HyStructParam (hConnect, &lStructType, &lParamd, &lPhraseId, &lUnitId,
          iFlag);
  iFlag = HYGETNEXT;
  if ( bOK )
    {
      ... do something with parameters returned
    }
  }

HyCommit ( hConnect );
```

HyStructPhrase

BOOL HyStructPhrase (*hConnect, lId, sName, iFlag*)

```
HYHAND    hConnect           /* HYDATA connection handle */
LPLONG    lId                /* Structure phrase id */
LPSTR     sName              /* Structure phrase name */
int       iFlag              /* Function control flag */
```

The **HyStructPhrase** function controls information concerning the pre-loaded information relating to hydraulic structure phrases (quantifiable parameters).

Parameter	Description
hConnect	HYDATA connection handle.
lId	Structure phrase id
sName	Name of hydraulic structure phrase
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first phrase type **HYGETNEXT** - Get next phrase type

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 62

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first type of hyraulic structure phrase. ***lId*** and ***sName*** are returned. Data are retrieved in ascending order of ***lId***. The **HyGetNext** function **cannot** be used with this type of data because the query is complex; use the **HYGETNEXT** flag with the current function (**HyStructPhrase**) to get remaining Cd types.

Note that this table is pre-loaded and cannot be edited by the user.

Example

```
BOOL      bOK;

int       iFlag;

long      lId;

char      sName [ 81 ];

HYHAND    hConnect;


/* Get hydraulic structure phrase types */

bOK = TRUE;
```

```
iFlag = HYGETINIT;

while ( bOK )
  {
  bOK = HyStructPhrase (hConnect, &lId, sName, iFlag);
  iFlag = HYGETNEXT;
  if ( bOK )
    {
      ... do something with phrase returned
    }
  }

HyCommit ( hConnect );
```

HyStructType

BOOL HyStructType (***hConnect, lId, sName, sGaugeFlag, sGateFlag, lCdTypeMask, sHydataFlag, iFlag***)

```
HYHAND    hConnect       /* HYDATA connection handle */
LPLONG    lId            /* Structure type id */
LPSTR     sName          /* Structure type name */
LPSTR     sGaugeFlag     /* Set to 'Y' if gauge is at structure */
LPSTR     sGateFlag      /* Set to 'Y' if there is a gate */
LPLONG    lCdTypeMask    /* Allowable Cd types as a mask */
LPSTR     sHydataFlag    /* Set to 'Y' if usable with HYDATA */
int       iFlag          /* Function control flag */
```

The **HyStructType** function controls information concerning the pre-loaded information on types of hydraulic structure.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Structure type id
sName	Name of hydraulic structure type
sGaugeFlag	Set to 'Y' to indicate a gauge board
sGateFlag	Set to 'Y' to indicate the structure has a gate
lCdTypeMask	Allowable Cd's for this type of structure or'ed together. Each instance of this type of structure can only have one type of allowable Cd.
sHydataFlag	Set to 'Y' to indicate this type of structure can be used with HYDATA.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first structure type **HYGETNEXT** - Get next structure type

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 60

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first type of hyraulic structure. ***lId, sName, sGaugeFlag, sGateFlag, lCdTypeMask*** and ***sHydataFlag*** are returned. Data are retrieved in ascending order of ***lId***. The **HyGetNext** function **cannot** be used with this type of data because the query is complex; use the **HYGETNEXT** flag with the current function (**HyStructType**) to get remaining structure types.

Note that this table is pre-loaded and cannot be edited by the user.

Example

```
BOOL      bOK;
```

```
int       iFlag;

long      lId, lCdTypeMask;

char      sName [ 81 ];
char      sGaugeFlag [ 2 ];
char      sGateFlag [ 2 ];
char      sHydataFlag [ 2 ];

HYHAND    hConnect;


/* Get hydraulic structure types */

bOK = TRUE;
iFlag = HYGETINIT;

while ( bOK )
  {
  bOK = HyStructType (hConnect, &lId, sName, sGaugeFlag, sGateFlag, lCdTypeMask,
          sHydataFlag, iFlag);
  iFlag = HYGETNEXT;
  if ( bOK )
    {
      ... do something with structure type
    }
  }

HyCommit ( hConnect );
```

HyStructure

```
BOOL HyStructure ( hConnect, lId, lpszName, lStationId, lTypeId, lTSId,
                   fDatum, lOrderNo, lCdTypeId, lCdId, iFlag )

HYHAND      hConnect          /* HYDATA connection handle */
LPLONG      lId               /* Structure id */
LPSTR       lpszName          /* Structure name */
LPLONG      lStationId        /* Station id */
LPLONG      lTypeId           /* Structure type id */
LPLONG      lTSId             /* Time series id */
double far * fDatum           /* Structure datum */
LPLONG      lOrderNo          /* Order number */
LPLONG      lCdTypeId         /* Cd type id */
LPLONG      lCdId             /* Cd id */
int         iFlag             /* Function control flag */
```

The **HyStructure** function controls information concerning hydraulic structures at a station.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Structure id
lpszName	Structure name
lStationId	Station id where structure is located
lTypeId	Structure type id
lTSId	Time series id for gate gauge on structure. Set to zero if no gate gauge on structure
fDatum	Datum of structure sill
lOrderNo	Order number of sequence number of structure at the station (eg 2 for 2nd structure in group).
lCdTypeId	The Cd type the user has chosen for use with this type of structure (the types allowed for any one strcuture type are found using the function **HyStructType**).
lCdId	A unique id that identifies the Cd definition data for the structure via the function **HyStructCd**.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first structure at a station **HYGETNEXT** - Get next structure at a station **HYUPDATE** - Update a structure **HYADD** - Add a new structure **HYDELETE** - Remove a structure

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 57

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first hydraulic structure at station specified in ***lStationId*** as ***lId, lpszName, lTypeId, lTSId, fDatum, lOrderNo, lCdTypeId*** and ***lCdId.*** Data are retrieved in ascending order of ***lOrderNo.*** The **HyGetNext** function can be used with this type of data to get all remaining structures at the station.

When the **HYUPDATE** flag is used ***lpszName, lTSId, fDatum, lOrderNo, lCdTypeId*** and ***lCdId*** are changed for the structure specified in ***lId.***

When the **HYADD** flag is used a new structure is added to the database. All parameters are required for this type of function call.

When the **HYDELETE** flag is used the structure specified by its id is removed from the database. ***lId*** must be specified for this call. Any other data associated with the structure (eg its gate level time series) is not removed; it is the responsibility of the application to make sure that it is.

Example

```
BOOL      bOK;

long      lId, lStationId, lTypeId, lTSId, lOrderNo, lCdTypeId, lCdId;

double    fDatum;

char      sName [ 81 ];

HYHAND    hConnect;


/* Add a new structure */

lId = 14L;
lStationId = 54L;
lTSId = 0L;
lTypeId = 21L;
lOrderNo = 1L;
lCdTypeId=128L
lCdId = 3L;
fDatum = 102.34;

lstrcpy ( sName, "Crump Weir" );

bOK = HyStructure ( hConnect, &lId, sName, &lStationId, &lTypeId, &lTSId,
&fDatum, &lOrderNo, &lCdTypeId, &lCdId, HYADD );

HyCommit ( hConnect );
```

HyTimeOut

BOOL HyTimeOut (*hConnect*)

```
HYHAND    hConnect          /* HYDATA connection handle */
```

The **HyTimeOut** function determines whether or not the failure of the previous function call was due to a database time out error. A time out error occurs when the data requested are in use by another user. All database function failures should be checked for this error and the user warned of the problem.

Parameter	Description
hConnect	HYDATA connection handle.

Returns

Returns TRUE if the last function failed due to a database time out error.

Export ordinal

DLL export ordinal: 13

Comments

Example

```
bOK =       HyGetApps ( hConnect, &lAppId, sAppName, &lAppType, sAppExe,
            HYGETINIT );
if ( !bOK )
  {
  if ( HyTimeOut ( hConnect ) )
    {
    wsprintf ( (LPSTR) sTmp, "Data in use by another user try later" );
    return FALSE;
    }
  else
    {
    wsprintf ( (LPSTR) sTmp, "Data not found" );
    return FALSE;
    }
  }

  return TRUE;
```

HyTSData

BOOL HyTSData (***hConnect, lpszTable, fReadTime, fData, lDataFlag, fDate, iFlag***)

```
HYHAND          hConnect          /* HYDATA connection handle */
LPSTR           lpszTable         /* Time series table name */
double far *    fReadTime         /* Date and time of reading */
double far *    fData             /* Data value */
LPLONG          lDataFlag         /* Data flag id */
double          fDate             /* Date for getting or deleting data */
int             iFlag             /* Function control flag */
```

The **HyTSData** function controls information concerning time series data readings. Data are retrieved either on a daily basis or sequentially for the whole series.

Parameter	Description
hConnect	HYDATA connection handle.
lpszTable	Time series data table name
fReadTime	Date and time of reading
fData	Data reading
lDataFlag	Data flag
fDate	Date (only) for getting and deleting data. If set to the constant **TSGETALL** when getting data, data for the whole series are retrieved in chronological order.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first reading **HYGETNEXT** - Get next reading **HYUPDATE** - Update reading **HYADD** - Add a new reading **HYDELETE** - Remove all readings in a day **HYINSERTINIT** - Insert the first of a block of data **HYINSERTNEXT** - Insert the subsequent readings **HYDELETEALL** - Delete all readings in a time series **HYGETINITDATE** - Retrieves all readings, on or after a specified date **HYGETINITDESC** - Retrives all readings in reverse order **HYDELETERANGE** - deletes a range of values

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 51

Comments

lpszTable must be specified on all types of call to define that table holding the time series.

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first reading in the day ***lDate,*** as ***fReadTime, fData*** and ***lDataFlag***. Data are retrieved in ascending order of time. If ***fData*** is set to the constant **TSGETALL** the whole time series is returned; if ***fData*** is set to a valid date only that data for the day is returned. The **HyGetNext** function can be used with this type of data to get all remaining data for the series/day.

HYGETINITDATE retrieves data from starting on or after the date specified in ***fDate*** to the end of the series. **HYGETINITDESC** retrieves data in reverse order starting with the first reading on or before ***fDate***.

When the **HYUPDATE** flag is used ***fData*** and ***lDataFlag*** are changed for the data reading specified in ***fReadTime***.

When the **HYADD** flag is used a new reading is added to the database table ***lpszTable***. ***fReadTime, fData*** and ***lDataFlag*** are also required for this type of function call.

When the **HYDELETE** flag is used all readings for the date given in ***fDate***. The **HYDELETEALL** flags deletes all readings in the time series. **HYDELETERANGE** deletes all values between ***fReadTime*** and ***fDate*** inclusive[*] .

A block of data can be inserted more efficiently than by repeated use of the **HYADD** flag by using the **HYINSERTINIT** flag for the first gauging and then calling the function **HyInsertNext** for all subsequent gaugings. All parameters must be supplied. For languages where the address of function parameters changes (eg SAL), the **HYINSERTNEXT** flag must be used with the function **HyTSData** (with all parameters supplied) rather than using the faster **HyInsertNext** function. The **HyInsertEnd** function must be called after the final insert to free resources associated with the insert.

Example

```
BOOL     bOK;

long     lDataFlagId;

double   fReadTime, fData;

char     sTable [ 32 ];

HYHAND   hConnect;


/* Add a new time series reading */

lstrcpy ( sTable, "TS23" );
fReadTime = 34354.5;
fData = 3674.234;
lDataFlagId = 0L;

bOK = HyTSData ( hConnect, sTable, &fReadTime, &fData, &lDataFlagId, 0.0, HYADD
);

HyCommit ( hConnect );
```

[*] Not yet implemented

HyTSDef

```
BOOL HyTSDef ( hConnect, lId, lpszName, lTypeId, lIntId, lpszTable,
               lObjTypeId, lObjId, fSDate, fEDate, fDatum, iFlag )

HYHAND      hConnect         /* HYDATA connection handle */
LPLONG      lId              /* Time series id */
LPSTR       lpszName         /* Time series name */
LPLONG      lTypeId          /* Time series type id */
LPLONG      lIntId           /* Interval id */
LPSTR       lpszTable        /* Time series data table name */
LPLONG      lObjTypeId       /* Object type id */
LPLONG      lObjId           /* Object id */
double far * fSDate          /* Start date of series */
double far * fEDate          /* End date of series */
double far * fDatum          /* Datum for water level readings */
int         iFlag            /* Function control flag */
```

The **HyTSDef** function controls information concerning the definition of time series.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Time series id
lpszName	Time series name
lTypeId	Time series type id. Use constant: TSWLUS TSWLDS TSGATE TSRESLEV TSFLOW TSRAIN TSGAUGING TSRATING TSSPOT TSLOCKAGE TSRESSTOR TSTEMPMIN TSTEMPMAX TSTEMPMEAN TSTEMPWETBULB TSTEMPDRYBULB TSEVAP TSSUNHOURS TSCLOUDCOVER TSWINDSPEED TSRHMIN TSRHMAX TSRHMEAN TSRADIATION TSNETRAD TSAIRPRESSURE
lIntId	Data interval id as managed by function **HyTSInts**
lpszTable	Time series data table name. Set to a NULL string if no table to be created (**HYADD** flag) or deleted (**HYDELETE** flag).
lObjTypeId	Object type id of object owning the time series
lObjId	Object id of object owning the time series
fSDate	Start date of the time series
fEDate	End date of the time series
fDatum	Level in metres to be added to all level readings when readings must be shown to common datum.
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first time series **HYGETNEXT** - Get next time series **HYSELECT** - Get details for specific time series id **HYSELECT2** - Get single time series **HYUPDATE** - Update time series name **HYADD** - Add a new time series **HYDELETE** - Remove a time series

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 49

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first category as ***lId, lpszName, lTypeId, lIntId, lpszTable, lObjTypeId, lObjId, fSDate, fEDate*** and ***fDatum***. Data are retrieved in ascending order of ***lId***. The function **HyGetNext** can be used with this type of data to get remaining table entries.

The **HYSELECT** flag returns ***lpszName, lTypeId, lIntId, lpszTable, lObjTypeId, lObjId, fSDate, fEDate*** and ***fDatum*** for a specified ***lId***.

The **HYSELECT2** flag returns ***lId, lpszName, lIntId, lpszTable, fSDate, fEDate*** and ***fDatum*** for a specified ***lObjTypeId, lObjId*** and ***lTypeId***.

When the **HYUPDATE** flag is used ***lpszName, lObjTypeId, lObjId, fSDate, fEDate*** and ***fDatum*** are changed for the type id specified in ***lId***.

When the **HYADD** flag is used a new time series definition is added to the database. All parameters are required for this type of function call. If no table is to be created to hold the data for the time series, ***lpszTable*** must be set to a NULL string ('\0'). No table is required for rating and spot gauging time series as these are all stored in a single table for each data type. For a standard time series, table names should be based on the time series id using the TS prefix (eg table name TS23 for time series id 23). For a gauging time series, table names should be based on the time series id using the GG prefix (eg table name GG3 for time series id 3). The use of these prefixes helps to aid the understanding of the database tables. Note that ***lTypeId*** determines whether a standard or a gauging time series data table is created.

When the **HYDELETE** flag is used the time is removed from the database. If there is a data table holding the time series, ***lpszTable*** must be set to the name of the data table to be deleted. In this case both the definition and the data table are removed. ***lId*** must also be specified for this call.

Example

```
BOOL     bOK;

long     lId, lTypeId, lIntId, lObjTypeId, lObjId;

double   fSDate, fEDate, fDatum;

char        sName [ 81 ], sTable [ 20 ];

HYHAND   hConnect;


/* Add a new time series definition */

lId = 12L;
lTypeId = TSWLUS;
lIntId = 11L;
lObjTypeId = 7L;
lObjId = 3L;
```

```
fSDate = 36000.0;
fEDate = 36300.0;
fDatum = 412.34;

lstrycpy ( sName, "New time series" );
lstrycpy ( sTable, "TS12" );

bOK = HyTSDef ( hConnect, &Id, sName, &lTypeId, &lIntId, sTable, &lObjTypeId,
&lObjId, &fSDate, &fEDate, HYADD );

HyCommit ( hConnect );
```

HyTSExt

BOOL HyTSExt (*hConnect, lId, lOrderNo, fMax, fMin, fMean, fMeanN, iFlag*)

```
HYHAND     hConnect          /* HYDATA connection handle */
LPLONG     lId               /* Time series id */
LPLONG     lOrderNo          /* Order number */
double far *  fMax           /* Maximum */
double far *  fMin           /* Minimum */
double far *  fMean          /* Mean */
double far *  fMeanN         /* N point mean */
int        iFlag             /* Function control flag */
```

The **HyTSExt** function controls information concerning time series extremes.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Time series id
lOrderNo	Order number of extreme (eg day number in year for daily data)
fMax	Maximum recorded for the date from the whole record
fMin	Minimum recorded for the date from the whole record
fMean	Mean recorded for the date from the whole record
fMeanN	N point running mean - mean value from whole record
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first extreme **HYGETNEXT** - Get next extreme **HYUPDATE** - Update extremes **HYADD** - Add a new extreme **HYDELETE** - Remove all extremes for a time series

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 52

Comments

Use the **HYGETINIT** flag on the first call to prepare the query for a specified ***lId*** and the function return details of the first extreme as ***lOrderNo, fMax, fMin, fMean*** and ***fMeanN***. Data are retrieved in ascending order of ***lOrderNo***. Use the **HYGETNEXT** flag to get all remaining data.

When the **HYUPDATE** flag is used ***fMax, fMin, fMean*** and ***fMeanN*** are changed for the specified ***lId*** and ***lOrderNo***.

When the **HYADD** flag is used a new extreme is added to the database. All parameters are both for this type of function call.

When the **HYDELETE** flag is used the whole set of extremes is removed from the database for the specified ***lId***.

Example

```
BOOL      bOK;

long      lId, lOrderNo;

double    fMax, fMin, fMean, fMeanN;

HYHAND    hConnect;


/* Add a new extreme */

lId = 23L;
lOrderNo = 365L;

fMax = 100.34;
fMin = 20.34;
fMean = 38.9;
fMeanN = 37.0;

bOK = HyTSExt ( hConnect, &lId, &lOrderNo, &fMax, &fMin, &fMean, &fMeanN, HYADD
);

HyCommit ( hConnect );
```

HyTSInts

BOOL HyTSInts (*hConnect, lId, lpszName, lInt, iFlag*)

```
HYHAND    hConnect           /* HYDATA connection handle */
LPLONG    lId                /* Time series interval id */
LPSTR     lpszName           /* Interval name */
LPLONG    lInt               /* Time series interval */
int       iFlag              /* Function control flag */
```

The **HyTSInts** function controls information concerning time series data intervals.

Parameter	Description
hConnect	HYDATA connection handle.
lId	Time series interval id
lpszName	Time series interval name
lInt	Data interval in seconds. Special values are: TSINTVAR = Variable time step series TSINTFIXIRR = Fixed irregular. One or more entries in the READ_TIME table defining the times in the day that readings occur. TSINTMONTH = Monthly data TSINTANNUAL = Annual data
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first interval **HYGETNEXT** - Get next interval **HYUPDATE** - Update interval name & interval **HYADD** - Add a new interval **HYDELETE** - Remove an interval

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 47

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first interval as ***lId***, ***lInt***, and ***lpszName***. Data are retrieved in ascending order of ***lId***. Since the retrieval is complex the **HyGetNext** function cannot be used with this type of data. Use the **HYGETNEXT** flag on **HyTSInts** to get all remaining data.

When the **HYUPDATE** flag is used ***lpszName*** and ***lInt*** are changed for the interval id specified in ***lId***. Applications must not change system defined time series intervals (ie ***lId*** MUST be negative).

When the **HYADD** flag is used a new category is added to the database. ***lId***, ***lInt***, and ***lpszName*** are all required for this type of function call. Applications must only add user defined time series intervals (ie ***lId*** MUST be negative).

When the **HYDELETE** flag is used the interval is removed from the database. Note that the present version of this function does not remove any data associated with this interval; it is currently the responsibility of the application to make sure that it is. ***lId*** must be specified for this call.

Example

```
BOOL      bOK;

long      lId, lInt;

HYHAND    hConnect;


/* Add a new time series category */

lId = -3L;
lInt = TSINTVAR;

bOK = HyTSInts ( hConnect, &lId, "New interval", &lInt, HYADD );

HyCommit ( hConnect );
```

HyTSReadTimes

```
BOOL HyTSReadTimes ( hConnect, lIntId, lSecs, iFlag )

HYHAND   hConnect           /* HYDATA connection handle */
LPLONG   lIntId             /* Time series interval id */
LPLONG   lSecs              /* Read time */
int      iFlag              /* Function control flag */
```

The **HyTSReadTimes** function controls information concerning time series reading times.

Parameter	Description
hConnect	HYDATA connection handle.
lIntId	Time series interval id
lSecs	Reading time as seconds into the day
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first reading time **HYGETNEXT** - Get next reading time **HYADD** - Add a new reading time **HYDELETE** - Remove a reading time

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 50

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first reading time for the time series data type specified in ***lIntId*** as ***lSecs***. Data are retrieved in ascending order of ***lSecs***. The **HyGetNext** function can be used with this type of data to get the remaining read times.

When the **HYADD** flag is used a new read time is added to the database. ***lIntId*** and ***lSecs*** are both required for this type of function call.

When the **HYDELETE** flag is used the read time is removed from the database for the specified ***lIntId*** and ***lSecs***.

Example

```
BOOL      bOK;

long      lIntId, lSecs;

HYHAND    hConnect;


/* Add a new time series reading time */
```

```
lIntId = -3L;
lSecs = 3600L;

bOK = HyTSReadTimes ( hConnect, &lIntId, &lSecs, HYADD );

HyCommit ( hConnect );
```

HyTSTypes

BOOL HyTSTypes (*hConnect, lTypeId, lpszName, lMeasType, lUnitId, iFlag*)

```
HYHAND   hConnect       /* HYDATA connection handle */
LPLONG   lTypeId        /* Time series type id */
LPSTR    lpszName       /* Type name */
LPLONG   lMeasType      /* Measurement type */
LPLONG   lInitId        /* Measurement unit id */
int      iFlag          /* Function control flag */
```

The **HyTSTypes** function controls information concerning time series types.

Parameter	Description
hConnect	HYDATA connection handle.
lTypeId	Time series type id
lpszName	Time series type name
lMeasType	Set to either the constant **MEASTYPEINST** to indicate the measurement type is instant (eg water level) or to **MEASTYPESUM** meaning the sum of readings over the last time period (eg rainfall)
lUnitId	Set to the measurement unit id appropriate for this type of data. Use constants: HYUNITMAP HYUNITELEV HYUNITDISTANCE HYUNITWATERLEVEL HYUNITFLOW HYUNITRAINFALL HYUNITRESSTOR HYUNITGATELEV HYUNITRESLEV HYUNITLOCKAGE HYUNITTEMP HYUNITEVAP HYUNITSUNHOUR HYUNITCLOUDCOV HYUNITWINDSPEED HYUNITRH HYUNITRADIATION HYUNITAIRPRESS
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first type **HYGETNEXT** - Get next type **HYUPDATE** - Update type **HYADD** - Add a new type **HYDELETE** - Remove a type

Returns

Returns TRUE if the call was successful, FALSE if not.

Export_ordinal

DLL export ordinal: 48

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and the function return details of the first type as ***lTypeId, lpszName, lMeasType*** and ***lUnitId.*** Data are retrieved in ascending order of ***lTypeId***. Since the retrieval is complex the **HyGetNext** function cannot be used with this type of data. Use the **HYGETNEXT** flag on **HyTSTypes** to get all remaining data.

When the **HYUPDATE** flag is used ***lpszName***, ***lMeasType*** and ***lUnitId*** are changed for the type id specified in ***lTypeId***. Applications must not change system defined time series data types (ie ***lTypeId*** MUST be negative).

When the **HYADD** flag is used a new time series data type is added to the database. ***lTypeId***, ***lMeasType***, ***lUnitId*** and ***lpszName*** are required for this type of function call. Applications must only add user defined time series data types (ie ***lTypeId*** MUST be negative).

When the **HYDELETE** flag is used the time series data type is removed from the database. Note that the present version of this function does not remove any data associated with this type; it is currently the responsibility of the application to make sure that it is. ***lTypeId*** must be specified for this call.

Example

```
BOOL      bOK;

long      lTypeId, lMeasType, lUnitId;

HYHAND    hConnect;


/* Add a new time series category */

lTypeId = -3L;
lMeasType = MEASTYPEINST;
lUnitId = HYUNITWATERLEVEL;

bOK = HyTSTypes ( hConnect, &lTypeId, "New type", &lMeasType, HYADD );

HyCommit ( hConnect );
```

HyUnits

BOOL HyUnits (*hConnect, lUnitId, lOrder, lpszUsage, lpszSI, lpszLocal, fMult, fConst, iDecPlace, iFlag*)

```
HYHAND       hConnect        /* HYDATA connection handle */
LPLONG       lUnitId         /* Unit id */
LPLONG       lOrder          /* Order in which units are retrieved from database
                             */
LPSTR        lpszUsage       /* Usage of unit */
LPSTR        lpszSI          /* Internal (SI) unit name */
LPSTR        lpszLocal       /* Local unit name */
float far *     fMult        /* Unit conversion multiply factor */
float far *     fConst       /* Unit conversion constant factor */
LPINT        iDecPlace       /* No. of decimal places for display */
int          iFlag           /* Function control flag */
```

The **HyGetUnits** function retrieves HYDATA units conversion information. The relationship between external and internal units is:

External Units = (Internal Units * Multiply) + Constant

Parameter	Description
hConnect	HYDATA connection handle.
lUnitId	Unit id
lpszUsage	Usage of the unit as a string.
lpszSI	The internal unit name (normally a SI unit).
lpszLocal	The local unit name
fMult	The unit conversion multiply factor
fConst	The unit conversion constant factor
iDecPlace	The number of decimal places for the display of the unit
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get first unit **HYGETNEXT** - Get next unit **HYUPDATE** - Updates a unit

Returns

This function returns TRUE if successful or FALSE if the request fails.

Export ordinal

DLL export ordinal: 14

Comments

HYDATA stores all its numeric data in its own internal units. Users can change units at any time to their own prefered external unit. Each application must always present numeric data in the form of external units and return to the database as internal units.

Use the **HYGETINIT** flag on the first call to prepare the query and return details of the first unit as ***lUnitId, lpszUsage, lpszSI, lpszLocal, fMult, fConst*** and

iDecPlace. The remaining units are retrieved by using **HyGetUnits** with the **HYGETNEXT** flag. The end of the units is signified by a return value of FALSE.

Note that due to the complex nature of this retrieval, **HyGetNext** cannot be used in conjunction with this function; the **HYGETNEXT** flag must be used with **HyGetUnits** to obtain a list of units.

When the **HYUPDATE** flag is used ***lpszLocal, fMult, fConst*** and ***lDecPlace*** are changed for the unit id specified in ***lUnitId***.

Example

```
BOOL      bOK;

int       iDecPlace;

long      lUnitId;

float          fMult, fConst;

char      sUsage [ 81 ];
char      sSI [ 81 ];
char      sLocal [ 81 ];
char      sTmp [ 386 ];

HYHAND    hConnect;

bOK =        HyGetUnits ( hConnect, &lUnitId, sUsage, sSi, sLocal, &fMult,
             &fConst, &iDecPlace, HYGETINIT );
sprintf ( sTmp, "Id %d Usage %s SI %s Local %s Mult %f Const %f Dec. Places %d",
             lUnitId, sUsage, sSi, sLocal, fMult, fConst, iDecPlace );

while ( bOK )
  {
  bOK =      HyGetUnits ( hConnect, &lUnitId, sUsage, sSi, sLocal, &fMult,
             &fConst, &iDecPlace, HYGETNEXT );
  sprintf ( sTmp, "Id %d Usage %s SI %s Local %s Mult %f Const %f Dec. Places
             %d", lUnitId, sUsage, sSi, sLocal, fMult, fConst, iDecPlace );
  }

HyCommit ( hConnect );
```

HyUpdateIndex

```
BOOL HyUpdateIndex ( hConnect, lpszIndex )

HYHAND    hConnect          /* HYDATA connection handle */
LPSTR     lpszIndex         /* HYDATA table index */
```

The **HyUpdateIndex** function updates the statistics on a HYDATA database table index.

Parameter	Description
hConnect	HYDATA connection handle.
lpszLocal	The name of the index to update. The prefix "SYSADM." is not required.

Returns

This function returns TRUE if successful or FALSE if the request fails.

Export ordinal

DLL export ordinal: 15

Comments

Statistics used for the efficient operation of the GUPTA SQLBase table index are updated by calling this function with the name of the index to be updated. Updating the statistics on an index can increase the performance of data retrievals which make use of that index. Indexes should be updated when a large amount of new data is added to a table.

Example

```
BOOL      bOK;
HYHAND    hConnect;

bOK =        HyUpdateIndex ( hConnect, "ATT_1" );

HyCommit ( hConnect );
```

HyUsers

BOOL HyUsers (*hConnect, iUserid, lpszUserName, lpszPassword, iAuth, iFlag*)

```
HYHAND   hConnect        /* HYDATA connection handle */
LPINT    iUserId         /* User id */
LPSTR    lpszUserName    /* User name */
LPSTR    lpszPassword    /* Users password */
LPINT    iAuth           /* User authority (privilege level) */
int      iFlag           /* Function control flag */
```

The **HyUsers** function controls information concerning HYDATA users.

Parameter	Description
hConnect	HYDATA connection handle.
iUserId	HYDATA user id
lpszUserName	HYDATA and database user name (max 8 characters)
lpszPassword	User HYDATA and database password
iAuth	The user's authority or privilege level (integer in the range 1 to 3)
iFlag	Function control flag which can take one of the following constant: **HYGETINIT** - Get user **HYGETNEXT** - Get next user **HYUPDATE** - Update user information **HYADD** - Add a new user to the database and HYDATA **HYDELETE** - Remove a user (database and HYDATA)

Returns

Returns TRUE if the call was successful, FALSE if not.

Export ordinal

DLL export ordinal: 19

Comments

Use the **HYGETINIT** flag on the first call to prepare the query and return details of the first user as ***iUserId, lpszUserName*** and ***iAuth***. The users applications are most efficiently retrieved by repeatedly calling **HyGetNext** and finally **HyGetEnd** (C and FORTRAN). For languages where the address of ***iUserId, lpszUserName*** or ***iAuth*** might change between calls, such as SAL, the remaining users must be retrieved using **HyUsers** with the **HYGETNEXT** flag. The end of the users is signified by a return value of FALSE. (***lpszPassword*** is not used.)

When the **HYUPDATE** flag is used ***iAuth*** is changed for the user id specified in ***iUserId***. (***lpszUserName*** and ***lpszPassword*** are not used.)

When the **HYADD** flag is used a new user is added to the database and to HYDATA. ***UserIid, lpszUserName, lpszPassword*** and ***iAuth*** are all required for this type of function call. Only user SYSADM can add a new user to the system. The application must supply a unique user id and database user name.

When the **HYDELETE** flag is used the user is removed from the database. Both ***UserIid*** and ***lpszUserName*** must be specified. (***lpszPassword*** and ***iAuth*** are not used.)

Example

```
BOOL      bOK;

int       iUserId, iAuth;

HYHAND    hConnect;


/* Add a new user */

iUserId = 53;
iAuth = 2;

bOK = HyUser (hConnect, &iUserId, "NEWUSER", "NEWPASS", &iAuth, HYADD);

HyCommit ( hConnect );
```

HyWarnMsg

```
void HyWarnMsg ( hConnect, hwndParent, lpszInfo )

HYHAND    hConnect        /* HYDATA connection handle */
HWND      hwndParent      /* Parent window handle */
LPSTR     lpszInfo        /* Warning information */
```

The **HyWarnMsg** function displays the standard HYDATA warning message box.

Parameter	Description
hConnect	HYDATA connection handle associated with this connection.
hwndParent	The parent window handle for the message box. Use (HWND) 0 if no parent window.
lpszInfo	A pointer to a string containing information to be displayed in the warning message box

Returns

There is no return value.

Export ordinal

DLL export ordinal: 17

Comments

This function should be used to display all HYDATA warning messages so that they appear consistent to the user.

Example

```
HYHAND   hConnect;

/* Display string 23 (string type 2) for application id 5 */

HyWarnMsg ( hConnect, (HWND) 0, HyGetString ( hConnect, 5, 2, 23L ) );
```

HyYesNoMsg

BOOL HyYesNoMsg (*hConnect, hwndParent, lpszInfo, bYesDef*)

```
HYHAND   hConnect      /* HYDATA connection handle */
HWND     hwndParent         /* Parent window handle */
LPSTR         lpszInfo      /* Information to display */
BOOL     bYesDef       /* TRUE if "Yes" is the default */
```

The **HyYesNoMsg** function displays the standard HYDATA information message box.

Parameter	Description
hConnect	HYDATA connection handle associated with this connection.
hwndParent	The parent window handle for the message box. Use (HWND) 0 if no parent window.
lpszInfo	A pointer to a string containing information to be displayed in the message box
bYesDef	Set to TRUE if "Yes" is the default push button or FALSE if "No" is the default push button

Returns

The function returns TRUE if "Yes" is selected or FALSE if "No is selected".

Export ordinal

DLL export ordinal: 22

Comments

This function should be used to display all HYDATA yes/no queries so that they appear consistent to the user.

Example

```
HYHAND   hConnect;

/* Display the question in string 16 (string type 3) for app. id 6 */

if ( HyYesNoMsg ( hConnect, (HWND) 0, HyGetString(hConnect,6,3,16L)))
  {
  /* Carry out the action if "Yes" is chosen */
  };
else
  {
  /* Carry out the action if "No" is chosen */
  };
```

2. DATABASE DESIGN

2.1 Structure

The design of the database tables is illustrated by Figures 1 and 2. The following list describes each of the database tables:

Table	**Description**
APPS	Hydata applications
ATT	Attributes
ATT_CHAR	Character type attributes
ATT_DATE	Station date type attributes
ATT_FLT	Station floating point attributes
ATT_INT	Station integer type attributes
ATT_LCHR	Station long character type attributes
ATT_PIC	Station picture attributes
BOUND_LOC	Boundary location data
CATCHMENT	River catchment definition
CAT_BOUND	Catchment boundary definition
DATA_FLAG	Time series data flags
EXTREMES	Time series extremes
GG*	River current meter gauging data
LANGUAGE	Languages
MAP_LINES	User defined lines and areas to be drawn on map
MAP_LINE_DATA	Data for map lines
MAP_STRINGS	User defined character strings to be drawn on map
OBJECT_ATT	Object attributes
OBJECT_TYPE	Object type definition
PICTURE	Pictures as compressed bitmaps
RATING	Rating equation definition
RAT_DATA	Data defining the rating equation
READ_TIME	Read times for data types with irregular read times within day
RIVER	River definition
RIVER_LOC	River location data
SPOT_GAUGING	Spot gaugings
STATION	List of HYDATA stations
STATION_TYPE	List of HYDATA stations types
STRINGS	Multi-language strings
STRUCTURE	Hydraulic structures
STRUCT_CD	Hydraulic structure cd data
STRUCT_CD_TYPE	Hydraulic structure cd types
STRUCT_DATA	Hydraulic structure definition data
STRUCT_PARAM	Hydraulic structure parameters
STRUCT_PHRASE	Hydraulic structure phrase (or definition parameter)
STRUCT_TYPE	Hydraulic structure types
TIME_SERIES	Time series data stored at each station
TS*	Time series data
TS_INT	Time series recording interval
TS_TYPE	Categories of time series data
UNIT	SI to local unit conversions
USER_INFO	User information

Details of the data held in each of the database tables is as follows:

Table	**Column**	**#**	**Type**	**Desription**
APPS	ID	1	INTEGER	Application id
	NAME	2	VARCHAR	Application name
	TYPE	3	INTEGER	Application type (1=core, 2=analysis)
	EXE	4	VARCHAR	
ATT	ID	1	INTEGER	Attribute id (+ve = pre-defined, -ve = user defined)
	NAME	2	VARCHAR	Name of attribute
	TYPE	3	SMALLINT	Att. type (1=char, 2=long char, 3=int, 4=float, 5=date, 6=pic)
ATT_CHAR	OBJECT_TYPE_ID	1	INTEGER	Object type id
	OBJECT_ID	2	INTEGER	Object id
	ATT_ID	3	INTEGER	Attribute id (+ve = pre-defined, -ve = user defined)
	VALUE	4	VARCHAR	Attribute value
ATT_DATE	OBJECT_TYPE_ID	1	INTEGER	Object type id
	OBJECT_ID	2	INTEGER	Object id
	ATT_ID	3	INTEGER	Attribute id (+ve = pre-defined, -ve = user defined)
	VALUE	4	TIMESTMP	Attribute value
ATT_FLT	OBJECT_TYPE_ID	1	INTEGER	Object type id
	OBJECT_ID	2	INTEGER	Object id
	ATT_ID	3	INTEGER	Attribute id (+ve = pre-defined, -ve = user defined)
	VALUE	4	FLOAT	Attribute value
ATT_INT	OBJECT_TYPE_ID	1	INTEGER	Object type id
	OBJECT_ID	2	INTEGER	Object id
	ATT_ID	3	INTEGER	Attribute id (+ve = pre-defined, -ve = user defined)
	VALUE	4	INTEGER	Attribute value
ATT_LCHR	OBJECT_TYPE_ID	1	INTEGER	Object type id
	OBJECT_ID	2	INTEGER	Object id
	ATT_ID	3	INTEGER	Attribute id (+ve = pre-defined, -ve = user defined)
	VALUE	4	LONGVAR	Attribute value
ATT_PIC	OBJECT_TYPE_ID	1	INTEGER	Object type id
	OBJECT_ID	2	INTEGER	Object id
	ATT_ID	3	INTEGER	Attribute id (+ve = pre-defined, -ve = user defined)
	VALUE	4	INTEGER	Attribute value (picture id)
BOUND_LOC	ID	1	INTEGER	Boundary location id
	X_COORD	2	FLOAT	X Co-ordinate for map
	Y_COORD	3	FLOAT	Y Co-ordinate for map
CATCHMENT	ID	1	INTEGER	Catchment id
	NAME	2	VARCHAR	Catchment name
	PARENT_ID	3	INTEGER	Parent catchment id (0 if no parent)
	STATION_ID	4	INTEGER	Station id of catchment gauging station
	RIVER_LOC_ID	5	INTEGER	River location id
	OUND_LOC_ID	6	INTEGER	Starting boundary location id
CAT_BOUND	CAT_ID	1	INTEGER	Catchment id
	BOUND_LOC_ID	2	INTEGER	Boundary location id
	NEXT_BOUND_LOC_ID	3	INTEGER	Next boundary location id
DATA_FLAG	ID	1	INTEGER	Data flag (includes -1 = start of gap, -2 = end of gap)
	NAME	2	VARCHAR	Description of data flag
EXTREMES	TIME_SERIES_ID	1	INTEGER	Time series id
	ORDER_NUM	2	INTEGER	Order number of point (day number 1-366 for daily data)
	MAX_VAL	3	FLOAT	Maximum value recorded
	MIN_VAL	4	FLOAT	Minimum value recorded
	MEAN_VAL	5	FLOAT	Mean of all values recorded
	MEAN_N_VAL	6	FLOAT	N day running mean - mean value
GG*	READ_TIME	1	TIMESTMP	Date and time of gauging
	LEVEL	2	FLOAT	Mean water level during gauging
	FLOW	3	FLOAT	Calculated discharge
	VELOCITY	4	FLOAT	Mean velocity across section (flow/area)
	RATING_NAME	5	VARCHAR	Rating anme or (? = not assigned, + = applies to all ratings
	COMMENTS	6	VARCHAR	Comments on the gauging
LANGUAGE	ID	1	INTEGER	Language id code
	NAME	2	VARCHAR	Language name
MAP_LINES	ID	1	INTEGER	Line id
	NAME	2	VARCHAR	Name of line
	VIS_LEV	3	INTEGER	Visibility level (0 - 100)
	THICKNESS	4	FLOAT	Thickness of line
	STYLE_ID	5	INTEGER	Line style id
	COLOUR_ID	6	INTEGER	Line and fill colour id
	FILL_STYLE_ID	7	INTEGER	Fill style id (0 = no fill)
Table	**Column**	**#**	**Type**	**Desription**
MAP_LINE_DATA	MAP_LINE_ID	1	INTEGER	Map line id
	ORDER_NUM	2	INTEGER	Order number of point
	X_COORD	3	FLOAT	Map line x co-ordinate
	Y_COORD	4	FLOAT	Map line y co-ordinate

MAP_STRINGS	ID	1	INTEGER	String id
	NAME	2	VARCHAR	Name to drawn on map
	VIS_LEV	3	INTEGER	Visibility level (0 - 100)
	X_COORD	4	FLOAT	X co-ordinate of lower left character of string
	Y_COORD	5	FLOAT	Y co-ordinate of lower left character of string
	WIDTH	6	FLOAT	Character width (in map internal units)
	ANGLE	7	FLOAT	Draw angle of string (0 - 360)
	SYMBOL_ID	8	INTEGER	Symbol to draw at start of string (at X,Y)
	COLOUR_ID	9	INTEGER	Colour of text
OBJECT_ATT	OBJECT_TYPE_ID	1	INTEGER	Object type id
	ATT_ID	2	INTEGER	Attribute id
OBJECT_TYPE	ID	1	INTEGER	Object type id
	NAME	2	VARCHAR	Object type name
PICTURE	ID	1	INTEGER	Picture id
	FORMAT	2	INTEGER	Format id of the bitmap
	WIDTH	3	INTEGER	Picture width
	HEIGHT	4	INTEGER	Picture height
	COMP_METH	5	INTEGER	Compression method (0=none)
	SIZE_MAX	6	INTEGER	Number of bytes in un-compressed form
	COLOURS	7	INTEGER	Number of colours in bit map
	PICTURE	8	LONGVAR	Picture data
RATING	ID	1	INTEGER	Rating id
	NAME	2	VARCHAR	Rating name
	TIME_SERIES_ID	3	INTEGER	Time series id
	RATING_TYPE_ID	4	INTEGER	Rating type id (1 = power, 2 = poly)
	S_DATE	5	TIMESTMP	Start date and time
	E_DATE	6	TIMESTMP	End date and time
	S_DAY	7	INTEGER	Start day within year (1-366)
	E_DAY	8	INTEGER	End day within year (1-366)
	COMMENTS	9	VARCHAR	Comments on rating
RAT_DATA	TIME_SERIES_ID	1	INTEGER	Time series id
	RATING_ID	2	INTEGER	Rating id
	PARAM_ID	3	SMALLINT	Paramater id (form of equation specific)
	VALUE	4	FLOAT	Parameter value
READ_TIME	TS_INT_ID	1	INTEGER	Time series interval id
	READ_TIME	2	INTEGER	Read time as seconds from start of day
RIVER	ID	1	INTEGER	River id
	NAME	2	VARCHAR	River name
	RIVER_LOC_ID	3	INTEGER	Furthest d/s river location id
RIVER_LOC	ID	1	INTEGER	Location id
	DS_ID	2	INTEGER	Location id immediately downstream
	US_ID	3	INTEGER	Location id immediately u/s on main channel
	X_COORD	4	FLOAT	X co-ordinate for map
	Y_COORD	5	FLOAT	Y co-ordinate for map
	ELEVATION	6	FLOAT	Elevation of location (metres)
	CHAINAGE	7	FLOAT	Chainage to next d/s location (metres)
SPOT_GAUGING	ID	1	INTEGER	Spot gauging id
	TIME_SERIES_ID	2	INTEGER	Time series id
	READ_TIME	3	TIMESTMP	Date and time of sport gauging
	DISCHARGE	4	FLOAT	Discharge
	PERCENTILE	5	FLOAT	Percentile from nearby main gauging station
	COMMENTS	6	VARCHAR	Comments on the gauging
STATION	ID	1	INTEGER	Station id
	NUMBERA	2	VARCHAR	Station number (primary)
	NUMBERB	3	VARCHAR	Station number (secondary)
	NAME	4	VARCHAR	Station name
	STATION_TYPE_ID	5	INTEGER	Station type id
	X_COORD	6	FLOAT	Station x co-ordinate
	Y_COORD	7	FLOAT	Station y co-ordinate
	ANGLE	8	FLOAT	Map draw angle for station name
	RIVER_LOC_ID	9	INTEGER	River location id (0 if not on river)
STATION_TYPE	ID	1	INTEGER	Station type id
	NAME	2	VARCHAR	Station type name
STRINGS	APP_ID	1	INTEGER	Application id
	LANG_ID	2	INTEGER	Language id (0 for generic strings
	TYPE_ID	3	INTEGER	String type id (1 = standard, 2 = error)
	STRING_ID	4	INTEGER	String id (non unique)
	STRING	5	VARCHAR	String text

Table	**Column**	**#**	**Type**	**Desription**
STRUCTURE	ID	1	INTEGER	Structure id
	NAME	2	VARCHAR	Structure name
	STATION_ID	3	INTEGER	Station id
	TYPE_ID	4	INTEGER	Structure type id
	TS_ID	5	INTEGER	Time series id for gate readings
	DATUM	6	FLOAT	Structure datum
	ORDER_NUM	7	INTEGER	Order number of structure within station group
	CD_TYPE_ID	8		
	CD_ID	9		

STRUCT_CD	ID	1	INTEGER	Structure cd type id
	X_VALUE	2	FLOAT	X value
	CD_VALUE	3	FLOAT	Cd value
STRUCT_CD_TYPE	ID	1	INTEGER	Structure cd type id
	TYPE_FLAG	2	VARCHAR	Y if more than one calibration value
	CONV_UNIT	3	INTEGER	Conversion unit to use
	MIN_VALUE	4	FLOAT	Min value for checking
	MAX_VALUE	5	FLOAT	Max value for checking
	NAME	6	VARCHAR	Structure cd type name
STRUCT_DATA	PARAM_ID	1	INTEGER	Hydraulic structure parameter id
	STRUCT_ID	2	INTEGER	Hydraulic structure id
	VALUE	3	FLOAT	Hydraulic structure parameter value
STRUCT_PARAM	STRUCT_TYPE_ID	1	INTEGER	Structure type id
	PARAM_ID	2	INTEGER	Structure parameter id (sequence number)
	PARAM_PHRASE_ID	3	INTEGER	Structure phrase id
	PARAM_UNIT_ID	4	INTEGER	Structure parameter unit id
STRUCT_PHRASE	ID	1	INTEGER	Structure phrase id
	PHRASE	2	VARCHAR	Structure phrase name
STRUCT_TYPE	ID	1	INTEGER	Structure type id
	NAME	2	VARCHAR	Structure type name
	GAUGE_FLAG	3	VARCHAR	Set to Y to indicate a gauge board
	GATE_FLAG	4	VARCHAR	Set to Y to indicate structure has a gate
	CD_TYPE_MASK	5	INTEGER	Ored mask of CD types for this structure type
	HYDATA_FLAG	6	VARCHAR	Structure type can be used in HYDATA (Y) or not (N)
TIME_SERIES	ID	1	INTEGER	Time series id
	NAME	2	VARCHAR	Time series name
	TS_TYPE_ID	3	INTEGER	Time series type id
	TS_INT_ID	4	INTEGER	Time series interval id
	TABLE_NAME	5	VARCHAR	Name of table holding data for this series
	OBJECT_TYPE_ID	6	INTEGER	Object type id to which the series is attached
	OBJECT_ID	7	INTEGER	Object id to which the series is attached
	S_DATE	8	TIMESTMP	Date and time of first reading in the series
	E_DATE	9	TIMESTMP	Date and time of last reading in the series
	DATUM	10	FLOAT	Datum in metres to be added to all level readings
TS*	READ_TIME	1	TIMESTMP	Date and time of reading
	VALUE	2	FLOAT	Data value
TS_INT	ID	1	INTEGER	Time series interval id
	NAME	2	VARCHAR	Name of time series interval
	INTERVAL	3	INTEGER	Data interval in seconds or(-1=variable,-2 from READ_TIME,-3 =)
TS_TYPE	ID	1	INTEGER	Data type id (-ve is user defined)
	NAME	2	VARCHAR	Name of data type
	MEAS_TYPE	3	INTEGER	Measurement type: 1 = instant, 2 = sum over last period
	UNIT_ID	4	INTEGER	Measurement unit id
UNIT	ID	1	INTEGER	Unit id
	ORDER_NO	2	INTEGER	Display order number
	USAGE	3	VARCHAR	Usage of unit
	NAME_SI	4	VARCHAR	SI name for unit
	NAME_LOCAL	5	VARCHAR	Local name for unit
	MULTIPLY	6	FLOAT	Multiply factor to convert to SI (applied before CONSTANT)
	CONSTANT	7	FLOAT	Constant to add to convert to SI (applied after FACTOR)
	DEC_PL	8	INTEGER	Number of decimal places for display
USER_INFO	ID	1	INTEGER	User id
	NAME	2	VARCHAR	User name a database user name
	AUTH	3	INTEGER	User authority level (1-3)

2.2 SQL scripts

An initial HYDATA database is created by running the SQL script contained in the file HY_CR40.SQL on an empty SQLBase database. HY_CR40.SQL creates all the tables required by HYDATA together with their indexes and load preliminary information into the database to enable HYDATA to run.

This SQL script should be run using the HYDBMAN utility program (a Windows application).

HYDATA tables may be removed from the database by running the SQL script HY_RM40.SQL via the HYDBMAN utility. Note that only the basic set of HYDATA tables created by HY_CR40.SQL are removed by this script. Any additional time series tables created via HYDATA itself are not removed.