DlisCreator Library

DlisCreator is a library for creating files in DLIS (RP66) format version 1. The library exposes extern functions with which user application creates DLIS file. Full list of exported functions see in appendix A. First group of functions work with DLIS file:

```
/* File handling */
extern "C" __declspec(dllexport) DLISTREEHANDLE CreateRoot(LPCSTR ssi)
extern "C" __declspec(dllexport) DLIS_RESULT Save(LPCSTR FileName)
```

`CreateRoot` function creates Storage Unit Label (SUL) and returns handle of the root of tree-shaped structure (see Fig.1) which will be filled with records by user application. Field `'Identifier'` of SUL is filled with `ssi` text, if `ssi` is NULL “Default Storage Set” is stored. After creating the root the application creates records and calls `Save` function which saves records in the specified file, closes it and deletes used variables in memory.

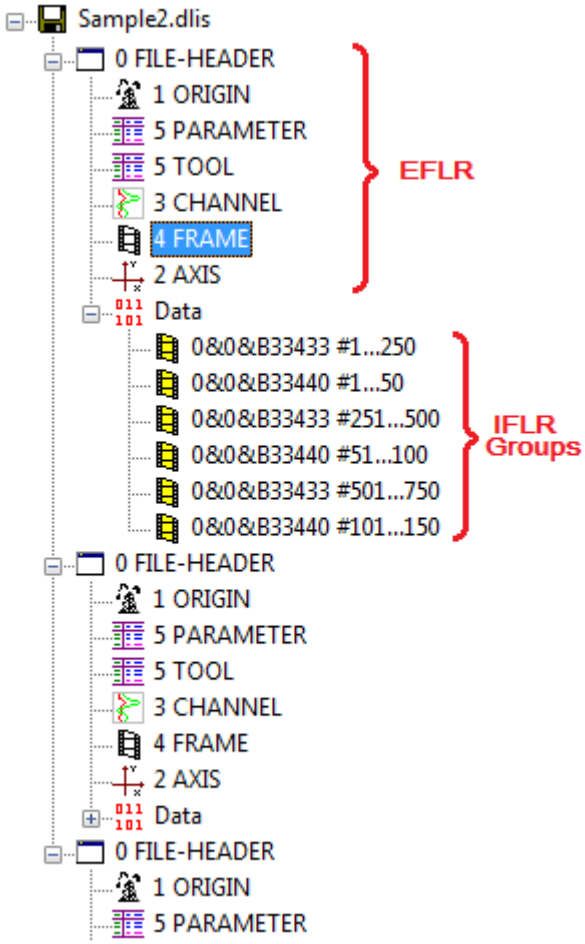

Figure 1

Each node in the tree represents a DLIS logical record described in RP66 specification http://w3.energistics.org/RP66/V1/Toc/main.html chapter 3.

The second group of exported functions is used to navigate through the tree:

```
/* Tree handling */
extern "C" __declspec(dllexport) DLIS_RESULT ChildCount(   // get count of children in the node
      /*in*/DLISTREEHANDLE node,
      /*out*/int* n)
extern "C" __declspec(dllexport) DLIS_RESULT Child(        // get n-th child of the node
      /*in*/DLISTREEHANDLE node, int n,
      /*out*/DLISTREEHANDLE* child)
extern "C" __declspec(dllexport) DLIS_RESULT Parent(       // get parent of the node
      /*in*/DLISTREEHANDLE node,
      /*out*/DLISTREEHANDLE* parent)
```

```
extern "C" __declspec(dllexport) DLIS_RESULT RecordType(   // get record type (Appendix 2, fig. A-2)
      /*in*/DLISTREEHANDLE node,
      /*out*/BYTE* bIFLR, BYTE* type, LPCSTR* set_type)
```

You pass into functions a node handle of type `DLISTREEHANDLE` and get handle of its child, root, etc. The functions return error code, full list of codes see in appendix B.

Function `RecordType` returns flag meaning is record IFLR or not, type of the record as it described in RP66 specification Appendix A, fig A-2, and set type in textual form. For example, Frame record (selected in Fig. 1) returns `bIFLR`=0, `type`=4 and `set_type`="FRAME". Some records return special value of type:
SUL (Storage Unit Label) returns -1
Encrypted record returns -2
Data (Group of IFLR records) returns -3

There are two kinds of records (see 3.1):

- an Explicitly Formatted Logical Record (EFLR)
- an Indirectly Formatted Logical Record (IFLR)

Functions for creating records

Creating EFLR

Each EFLR encapsulates a table of information. The rows of the table represent *Objects*, and the columns of the table represent *Attributes* of the *Objects*. At the beginning of the table there is a *Template* that defines the columns.

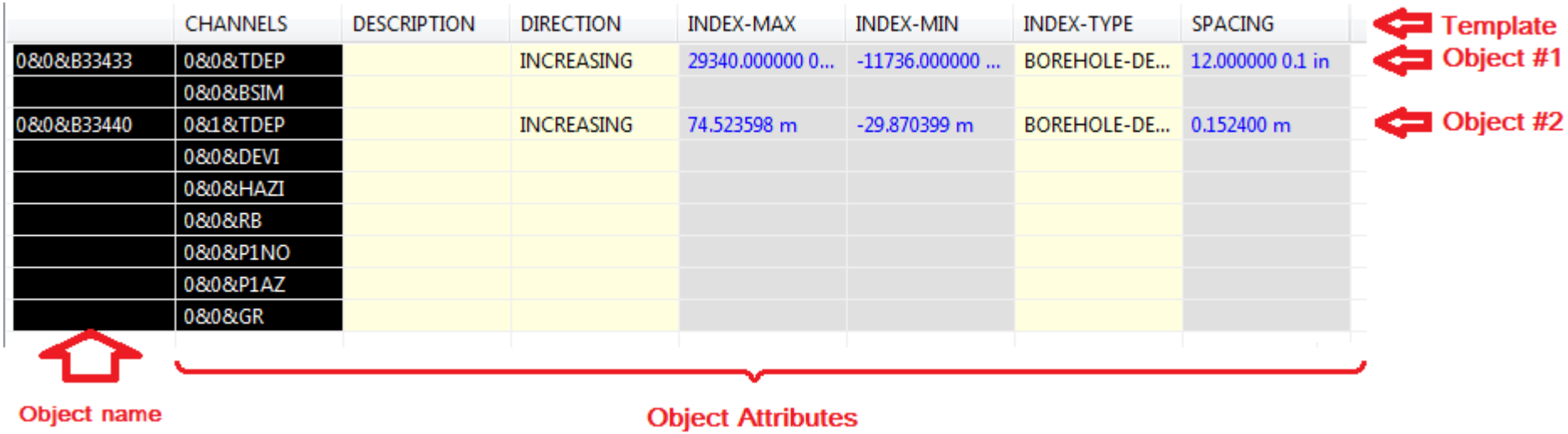

Figure 2

The following functions create EFLR records, their templates and objects:

```
extern "C" __declspec(dllexport) DLIS_RESULT CreateEFLR(DLISTREEHANDLE node, BYTE RecType, LPCSTR SetType, LPCSTR SetName, DLISTREEHANDLE* child)
extern "C" __declspec(dllexport) DLIS_RESULT CreateStockEFLR(DLISTREEHANDLE node, WORD StockSetType, LPCSTR SetName, DLISTREEHANDLE* child)
extern "C" __declspec(dllexport) DLIS_RESULT CreateTemplateItem(DLISTREEHANDLE node, REPR_CODE rc, LPCSTR TemplateItem)
extern "C" __declspec(dllexport) DLIS_RESULT FindTemplateItem(DLISTREEHANDLE node, LPCSTR name, int *i)
extern "C" __declspec(dllexport) DLIS_RESULT DeleteTemplateItem(DLISTREEHANDLE node, LPCSTR name)
extern "C" __declspec(dllexport) DLIS_RESULT CreateObject(DLISTREEHANDLE node, DLIS_OBNAME* objname, DLISOBJECTHANDLE* hobj)
```

Function `CreateEFLR` creates EFLR records with specified Record Type, Set Type and Set Name and adds it to specified tree node.

Function `CreateStockEFLR` creates EFLR records by predefined Stock Set Type and creates default template in it. List of Stock Set Type and their templates see in appendix C.

You can create objects and template items in any order. When a template item is created, corresponding attributes are created in all objects with ABSATR flags.

After the objects are created, the application fills them with attributes with the following functions:

```
extern "C" __declspec(dllexport) DLIS_RESULT SetAttribute(DLISOBJECTHANDLE hobj, LPCSTR attr_name,
      LPCSTR units, DLISvalue* val, int* indx)
extern "C" __declspec(dllexport) DLIS_RESULT SetAttributeByIndex(DLISOBJECTHANDLE hobj, int i, LPCSTR
      units, DLISvalue* val)
extern "C" __declspec(dllexport) DLIS_RESULT SetTextAttribute(DLISOBJECTHANDLE hobj, LPCSTR attr_name,
      LPCSTR units, LPCSTR str, REPR_CODE rc, int* indx)
extern "C" __declspec(dllexport) DLIS_RESULT SetTextAttributeByIndex(DLISOBJECTHANDLE hobj, int i,
      LPCSTR units, LPCSTR str, REPR_CODE rc)
extern "C" __declspec(dllexport) DLIS_RESULT SetIntAttribute(DLISOBJECTHANDLE hobj, LPCSTR attr_name,
      LPCSTR units, DWORD n, REPR_CODE rc, int* indx)
extern "C" __declspec(dllexport) DLIS_RESULT SetIntAttributeByIndex(DLISOBJECTHANDLE hobj, int i,
      LPCSTR units, DWORD n, REPR_CODE rc)
extern "C" __declspec(dllexport) DLIS_RESULT SetFloatAttribute(DLISOBJECTHANDLE hobj, LPCSTR attr_name,
      LPCSTR units, float f, int* indx)
extern "C" __declspec(dllexport) DLIS_RESULT SetFloatAttributeByIndex(DLISOBJECTHANDLE hobj, int i,
      LPCSTR units, float f)
extern "C" __declspec(dllexport) DLIS_RESULT SetDateAttribute(DLISOBJECTHANDLE hobj, LPCSTR attr_name,
      DLISdate* date, int* indx)
extern "C" __declspec(dllexport) DLIS_RESULT SetDateAttributeByIndex(DLISOBJECTHANDLE hobj, int i,
      DLISdate* date)
extern "C" __declspec(dllexport) DLIS_RESULT SetOBNAMEAttribute(DLISOBJECTHANDLE hobj, LPCSTR
      attr_name, DLIS_OBNAME* obname, int* indx)
extern "C" __declspec(dllexport) DLIS_RESULT SetOBNAMEAttributeByIndex(DLISOBJECTHANDLE hobj, int i,
      DLIS_OBNAME* obname)
extern "C" __declspec(dllexport) DLIS_RESULT SetOBJREFAttribute(DLISOBJECTHANDLE hobj, LPCSTR
      attr_name, DLIS_OBJREF* objref, int* indx)
extern "C" __declspec(dllexport) DLIS_RESULT SetOBJREFAttributeByIndex(DLISOBJECTHANDLE hobj, int i,
      DLIS_OBJREF* objref)
extern "C" __declspec(dllexport) DLIS_RESULT SetATTREFAttribute(DLISOBJECTHANDLE hobj, LPCSTR
      attr_name, DLIS_ATTREF* attref, int* indx)
extern "C" __declspec(dllexport) DLIS_RESULT SetATTREFAttributeByIndex(DLISOBJECTHANDLE hobj, int i,
      DLIS_ATTREF* attref)
```

Each function has two forms for setting attribute by name or by index. If you set attribute by name its index is returned in last parameter. It can be used in following calls for setting that attribute by index. Multiple calls for the same object with same template add multiple attributes. For example, CHANNELS attribute of FRAME object is filled with channels names. Functions `SetAttribute` and `SetAttributeByIndex` can set attribute with any representation code. The rest functions provide an easy way to set attributes with frequently used representation codes. `SetTextAttribute` and `SetTextAttributeByIndex` set attributes with codes IDENT, ASCII and UNITS. Note: these functions do not check the text for restrictions imposed by DLIS specification when IDENT or UNITS representation codes are used. `SetIntAttribute` and `SetIntAttributeByIndex` set SSHORT, SNORM, SLONG, USHORT, UNORM, ULONG, UVARI and STATUS. The rest functions set FSINGL, DTIME, OBNAME, OBJREF and ATTREF respectively. If representation code set by function does not correspond to that in template, DLIS_WRONG_REPR_CODE is returned.

Creating IFLR

Before creating IFLR the application shall create Data Group logical record as child of FHLR record. This record shall be last child in FHLR node. Then it shall create IFLR Group records corresponding to each Frame used. Note that Data Group and IFLR Group records are not a part of DLIS format, they are created for convenient navigation through the tree and fast access to IFLR records. Data Group is node containing all IFLRs of current logical file.

Note that number of IFLR Groups created by user application is equal to number of frames. If the application tries to create IFLR Group with Frame Name which already exists in Data Group, the function returns handle to the existing IFLR Group. When the application saves data to the file, DLIS Creator divides IFLR records into pieces and writes in alternating order.

The following functions are used for IFLR records creating:

```
extern "C" __declspec(dllexport) DLIS_RESULT CreateDataGroup(DLISTREEHANDLE node, DLISTREEHANDLE*
      data_group)
extern "C" __declspec(dllexport) DLIS_RESULT CreateIflrGroup(DLISTREEHANDLE node, DLIS_OBNAME*
      frame_name, DLISTREEHANDLE* iflr_group)
```

```
extern "C" __declspec(dllexport) DLIS_RESULT CreateIFLR(DLISTREEHANDLE node, DLISIFLRHANDLE* hiflr)
```

After an IFLR record is created, it must be populated with values. The following functions are used:

```
extern "C" __declspec(dllexport) DLIS_RESULT SetValue(DLISIFLRHANDLE hiflr, DLISvalue* v)
extern "C" __declspec(dllexport) DLIS_RESULT SetFloat(DLISIFLRHANDLE hiflr, float v)
extern "C" __declspec(dllexport) DLIS_RESULT SetDouble(DLISIFLRHANDLE hiflr, double v)
extern "C" __declspec(dllexport) DLIS_RESULT SetInt(DLISIFLRHANDLE hiflr, DWORD v)
```

`SetValue` function can write value with any representation code, other functions are used for frequently used codes. `SetFloat` is used for writing FSINGL, `SetDouble` – FDOUBL, `SetInt` – for SSHORT, SNORM, SLONG, USHORT, UNORM, ULONG and UVARI.

The values are filled in order listed in CHANNELS attribute of FRAME object. If a channel has dimension more than 1, Set*** function shall be called for each value of the channel. For correct writing of multidimensional channels see chapter 4.4.3 of http://w3.energistics.org/RP66/V1/Toc/main.html. Test return values to check correctness of values filling. When last value is written, DLIS_SUCCESS is returned; if not last value is written, DLIS_IFLR_NOT_FILLED is returned. This is not error, it is information message only. If representation code of the written value does not correspond to that in CHANNELS sequence, DLIS_WRONG_REPR_CODE is returned. If you call Set*** functions when IFLR is already filled, DLIS_WRONG_INDEX is returned.

Encrypted records handling

DLIS file may contain encrypted records. If customer provides encryption library, DLIS creator can encrypt records. The following functions are used to mark records to encrypt:

```
extern "C" __declspec(dllexport) DLIS_RESULT EncryptEFLR(DLISTREEHANDLE node, WORD ProdCode)
extern "C" __declspec(dllexport) DLIS_RESULT EncryptIFLR(DLISIFLRHANDLE hiflr, WORD ProdCode)
```

Note that these functions do not encrypt the record, but mark them only, encryption will be done when saving into file will be executed.

The user-supplied encryption library shall meet the following requirements:

- its name shall be 'encr###.dll' where ### - Producer's Company Code from Logical Record Segment Encryption Packet (LRSEP) in decimal form (no leading zeroes);
- it shall be located in the same directory where DlisCreator.dll is;
- it shall expose function 'Encrypt' with the following prototype:

```
extern "C" __declspec(dllexport) LRSEP* Encrypt(const BYTE* src_rec, WORD src_ln, BYTE** encrypted_rec,
      WORD* encrypted_ln)
```

where `src_rec` – pointer to source record;
`src_ln` – length of source record;
`encrypted_rec` – pointer to inner buffer where the library created encrypted record;
`encrypted_ln` – pointer to the length of encrypted buffer;

The function encrypts source record, places encrypted record to the inner buffer, writes pointer to it to `encrypted_rec`, its length to `encrypted_ln` and returns pointer to Logical Record Segment Encryption Packet (LRSEP).

Length of the encrypted packed shall be even. If it is odd, it shall be increased to 1 and pad byte 1 shall be placed to the end of the encrypted buffer. If encrypted record is larger than source, number of pad bytes shall be placed to the end of the encrypted buffer. This number shall be encrypted.

Appendix A

List of exported functions.

```
/* File handling */
typedef DLISTREEHANDLE (*CREATEROOT_FPTR)(LPCSTR ssi);
typedef DLIS_RESULT (*SAVE_FPTR)(const char* filename);

/* Tree handling */
typedef DLIS_RESULT (*CHILDCOUNT_FPTR)(            // get count of childs in the node
       /*in*/DLISTREEHANDLE node,
       /*out*/int* cnt);
typedef DLIS_RESULT (*CHILD_FPTR)(                 // get n-th child of the node
       /*in*/DLISTREEHANDLE node, int n,
       /*out*/DLISTREEHANDLE* child);
typedef DLIS_RESULT (*PARENT_FPTR)(                // get parent of the node
       /*in*/DLISTREEHANDLE node,
       /*out*/DLISTREEHANDLE* parent);
typedef DLIS_RESULT (*RECORDTYPE_FPTR)(            // get record type (Appendix 2, fig. A-2)
       /*in*/DLISTREEHANDLE node,
       /*out*/BYTE* bIFLR, BYTE* type, LPCSTR* set_type);

/* Records creating */
typedef DLIS_RESULT (*CREATE_EFLR_FPTR)(           // Create an EFLR record
    /*in*/DLISTREEHANDLE node, BYTE RecType, LPCSTR SetType, LPCSTR SetName,
    /*out*/DLISTREEHANDLE* child);
typedef DLIS_RESULT (*CREATE_STOCKEFLR_FPTR)(      // Create an EFLR record from stock
    /*in*/DLISTREEHANDLE node, WORD StockSetType, LPCSTR SetName,
    /*out*/DLISTREEHANDLE* child);
typedef DLIS_RESULT (*CREATE_TEMPLATEITEM_FPTR)(   // Create template item in the EFLR record
    /*in*/DLISTREEHANDLE node, REPR_CODE rc, LPCSTR TemplateItem);
typedef DLIS_RESULT (*FIND_TEMPLATEITEM_FPTR)(     // find template item by name
    /*in*/DLISTREEHANDLE node, LPCSTR name,
    /*out*/int* i);
typedef DLIS_RESULT (*DELETE_TEMPLATEITEM_FPTR)(   // delete template item by name
    /*in*/DLISTREEHANDLE node, LPCSTR name);
typedef DLIS_RESULT (*CREATE_OBJECT_FPTR)(         // Create an Object in the EFLR record
    /*in*/DLISTREEHANDLE node, DLIS_OBNAME* objname,
    /*out*/DLISOBJECTHANDLE* hobj);

/* Attributes handling */
typedef DLIS_RESULT (*SET_ATTRIBUTE_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, LPCSTR attr_name, LPCSTR units, DLISvalue* val,
    /*out*/int* i);
typedef DLIS_RESULT (*SET_ATTRIBUTE_BYINDX_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, int i, LPCSTR units, DLISvalue* val);

typedef DLIS_RESULT (*SET_TXT_ATTRIBUTE_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, LPCSTR attr_name, LPCSTR units, LPCSTR str, REPR_CODE rc,
    /*out*/int* i);
typedef DLIS_RESULT (*SET_TXT_ATTRIBUTE_BYINDX_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, int i, LPCSTR units, LPCSTR str, REPR_CODE rc);

typedef DLIS_RESULT (*SET_INT_ATTRIBUTE_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, LPCSTR attr_name, LPCSTR units, DWORD n, REPR_CODE rc,
    /*out*/int* i);
typedef DLIS_RESULT (*SET_INT_ATTRIBUTE_BYINDX_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, int i, LPCSTR units, DWORD n, REPR_CODE rc);

typedef DLIS_RESULT (*SET_FLT_ATTRIBUTE_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, LPCSTR attr_name, LPCSTR units, float n,
    /*out*/int* i);
typedef DLIS_RESULT (*SET_FLT_ATTRIBUTE_BYINDX_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, int i, LPCSTR units, float n);

typedef DLIS_RESULT (*SET_DATE_ATTRIBUTE_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, LPCSTR attr_name, DLISdate* date,
    /*out*/int* i);
typedef DLIS_RESULT (*SET_DATE_ATTRIBUTE_BYINDX_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, int i, DLISdate* date);

typedef DLIS_RESULT (*SET_OBNAME_ATTRIBUTE_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, LPCSTR attr_name, DLIS_OBNAME* obname,
    /*out*/int* i);
typedef DLIS_RESULT (*SET_OBNAME_ATTRIBUTE_BYINDX_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, int i, DLIS_OBNAME* obname);

typedef DLIS_RESULT (*SET_OBJREF_ATTRIBUTE_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, LPCSTR attr_name, DLIS_OBJREF* obname,
    /*out*/int* i);
```

```
typedef DLIS_RESULT (*SET_OBJREF_ATTRIBUTE_BYINDX_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, int i, DLIS_OBJREF* obname);

typedef DLIS_RESULT (*SET_ATTREF_ATTRIBUTE_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, LPCSTR attr_name, DLIS_ATTREF* obname,
    /*out*/int* i);
typedef DLIS_RESULT (*SET_ATTREF_ATTRIBUTE_BYINDX_FPTR)(
    /*in*/DLISOBJECTHANDLE hobj, int i, DLIS_ATTREF* obname);

/* IFLR handling */
typedef DLIS_RESULT (*CREATE_DATAGROUP)(
    /*in*/DLISTREEHANDLE node,
    /*out*/DLISTREEHANDLE* data_group);
typedef DLIS_RESULT (*CREATE_IFLRGROUP)(
    /*in*/DLISTREEHANDLE node, DLIS_OBNAME* frame_name,
    /*out*/DLISTREEHANDLE* iflr_group);
typedef DLIS_RESULT (*CREATE_IFLR)(
    /*in*/DLISTREEHANDLE node,
    /*out*/DLISIFLRHANDLE* hiflr);

typedef DLIS_RESULT (*SET_VALUE)(
    /*in*/DLISIFLRHANDLE hiflr, DLISvalue* v);
typedef DLIS_RESULT (*SET_FLOAT)(
    /*in*/DLISIFLRHANDLE hiflr, float v);
typedef DLIS_RESULT (*SET_DOUBLE)(
    /*in*/DLISIFLRHANDLE hiflr, double v);
typedef DLIS_RESULT (*SET_INT)(
    /*in*/DLISIFLRHANDLE hiflr, DWORD v);
```

Appendix B

List of error codes.

DLIS_SUCCESS	Successful completion
DLIS_FILE_ERROR	Error on opening or reading DLIS file; call GetLastError() to get system error code
DLIS_TMP_FILE_ERROR	Error in temporary file; call GetLastError() to get system error code
DLIS_FORMAT_ERROR	Common format error in DLIS file
DLIS_WRONG_HANDLE	Wrong handle
DLIS_WRONG_INDEX	Index is beyond array
DLIS_NOT_FOUND	Searched object not found
DLIS_NOT_EFLR	Node is not EFLR
DLIS_NOT_DATAGROUP	Node is not Data Group
DLIS_NOT_IFLRGROUP	Node is not IFLR Group or its type is not FDATA
DLIS_NO_FRAME	Logical file does not have FRAME record
DLIS_NO_CHANNEL	Logical file does not have CHANNEL record
DLIS_NO_CHANNELS_IN_FRAME	FRAME object does not have CHANNELS attribute
DLIS_CHANNEL_NOT_FOUND	A channel object from FRAME record not found in CHANNEL record
DLIS_REPR_CODE_NOT_FOUND	REPRESENTATION-CODE attribute not found in CHANNEL object
DLIS_NO_IFLR_NB	IFLR with requested sequential number not found
DLIS_VERS_NOT_SUPPORTED	DLIS version is not supported
DLIS_WRONG_REPR_CODE	Wrong representation code
DLIS_IFLR_NOT_FILLED	Value written in IFLR is not last
DLIS_NOT_LICENSED	Limited functionality of non-licensed library

Appendix C

List of stock set types.

Set name	Default template items created
SST_FILE_HEADER	SEQUENCE-NUMBER, ID
SST_ORIGIN	FILE-ID, FILE-SET-NAM, FILE-SET-NUMBER, FILE-NUMBER, FILE-TYPE, PRODUCT, VERSION, PROGRAMS, CREATION-TIME, ORDER-NUMBER, DESCENT-NUMBER, RUN-NUMBER, WELL-ID, WELL-NAME, FIELD-NAME, PRODUCER-CODE, PRODUCER-NAME, COMPANY, NAME-SPACE-NAME, NAME-SPACE-VERSION
SST_WELL_REFERENCE	PERMANENT-DATUM, VERTICAL-ZERO, PERMANENT-DATUM-ELEVATION, ABOVE-PERMANENT-DATUM, MAGNETIC-DECLINATION, COORDINATE-1-NAME, COORDINATE-1-VALUE, COORDINATE-2-NAME, COORDINATE-2-VALUE, COORDINATE-3-NAME, COORDINATE-3-VALUE
SST_AXIS	AXIS-ID, COORDINATES, SPACING
SST_CHANNEL	LONG-NAME, PROPERTIES, REPRESENTATION-CODE, UNITS, DIMENSION, ELEMENT-LIMIT, AXIS, SOURCE
SST_FRAME	DESCRIPTION, CHANNELS, INDEX-TYPE, DIRECTION, SPACING, ENCRYPTED, INDEX-MIN, INDEX-MAX
SST_PATH	FRAME-TYPE, WELL-REFERENCE-POINT, VALUE, BOREHOLE-DEPTH, VERTICAL-DEPTH, RADIAL-DRIFT, ANGULAR-DRIFT, TIME, DEPTH-OFFSET, MEASURE-POINT-OFFSET, TOOL-ZERO-OFFSET
SST_CALIBRATION	CALIBRATED-CHANNELS, UNCALIBRATED-CHANNELS, COEFFICIENTS, MEASUREMENTS, PARAMETERS, METHOD
SST_CALIBRATION_COEFFICIENT	LABEL, COEFFICIENTS, PREFERENCES, PLUS-TOLERANCES, MINUS-TOLERANCES
SST_CALIBRATION_MEASUREMENT	PHASE, MEASUREMENT-SOURCE, TYPE, DIMENSION, AXIS, MEASUREMENT, SAMPLE-COUNT, MAXIMUM-DEVIATION, STANDARD-DEVIATION, BEGIN-TIME, DURATION, REFERENCE, STANDARD, PLUS-TOLERANCES, MINUS-TOLERANCES
SST_COMPUTATION	LONG-NAME, PROPERTIES, DIMENSION, AXIS, ZONES, VALUES, SOURCE
SST_EQUIPMENT	TRADEMARK-NAME, STATUS, TYPE, SERIAL-NUMBER, LOCATION, HEIGHT, LENGTH, MINIMUM-DIAMETER, MAXIMUM-DIAMETER, VOLUME, WEIGHT, HOLE-SIZE, PRESSURE, TEMPERATURE, VERTICAL-DEPTH, RADIAL-DRIFT, ANGULAR-DIAMETER
SST_GROUP	DESCRIPTION, OBJECT-TYPE, OBJECT-LIST, GROUP-LIST
SST_PARAMETER	LONG-NAME, VALUES, DIMENSION, AXIS, ZONES
SST_PROCESS	DESCRIPTION, TRADEMARK-NAME, VERSION, PROPERTIES, STATUS, INPUT-CHANNELS, OUTPUT-CHANNELS, INPUT-COMPUTATIONS, OUTPUT-COMPUTATIONS, PARAMETERS, COMMENTS
SST_SPLICE	OUTPUT-CHANNELS, INPUT-CHANNELS, ZONES
SST_TOOL	DESCRIPTION, TRADEMARK-NAME, GENERIC-NAME, PARTS, STATUS, CHANNELS, PARAMETERS
SST_ZONE	DESCRIPTION, DOMAIN, MAXIMUM, MINIMUM
SST_COMMENT	TEXT
SST_NO_FORMAT	CONSUMER-NAME, DESCRIPTION
SST_LONG_NAME	GENERAL-MODIFIER, QUANTITY, QUANTITY-MODIFIER, ALTERED-FORM, ENTITY, ENTITY-MODIFIER, ENTITY-NUMBER, ENTITY-PART, ENTITY-PART-NUMBER, GENERIC-

	SOURCE, SOURCE-PART, SOURCE-PART-NUMBER, CONDITIONS, STANDARD-SYMBOL, PRIVATE-SYMBOL