ISIS Application Program Interface
ISIS_DLL User's Manual

Preliminary Version

BIREME, São Paulo, July 2001

Table of Contents

Chapter I. Introduction

This manual describes the use of ISIS Application Program Interface (API) in the form of a Dynamic Link Library (ISIS_DLL).

ISIS_DLL is an implementation of the ISIS Application Program Interface for the MS-Windows and Linux Operating Systems. BIREME/PAHO/WHO and UNESCO developed it.

ISIS_DLL is a tool for the development of ISIS applications under MS Windows 95, MS Windows 98, Windows NT, MS Windows 2000 and Linux graphic user interface, for 32 bits platforms. It may be called from any application written in Visual Basic, Delphi, C, C++, Java, Power Builder or any other language compiler, which supports DLL calls.

ISIS_DLL is fully compatible with the standard ISIS[1] systems, versions 3.07 for DOS and 1.0 for Windows. Therefore, applications using ISIS_DLL functions may coexist with other applications using the standard ISIS software.

Although a summary of the main features of ISIS is given in this manual, users may want to refer to the *CDS/ISIS Reference Manual*[2], published by UNESCO, particularly for a full description of such topics as the Formatting language, Field Select Tables and Search language, as well as to the READ.ME file published with the latest release of the software.

1. Manual Structure

This manual consists of six chapters and three annexes:

- Chapter I introduces the manual and installation options;
- Chapter II presents an overview of ISIS_DLL concepts, design and functionalities. It also explains the functions available to describe ISIS related entities;
- Chapter III describes the Master File data structures and the related input and output functions;

[1] Throughout this manual *Standard ISIS* refers to the CDS/ISIS software as distributed by UNESCO.

[2] Mini-micro CDS/ISIS, REFERENCE MANUAL (Version 2.3), UNESCO, Paris, 1989

- Chapter IV describes the loading and updating of inverted files in concurrent multi-user operations and related functions;
- Chapter V describes ISIS_DLL functions related to the Inverted File term dictionary and search processing;
- Chapter VI contains the alphabetic reference manual to ISIS_DLL functions;
- Annex 1 presents a list of ISIS_DLL data definitions and structures;
- Annex 2 presents a list of ISIS_DLL functions prototypes for C, Visual Basic, Delphi and Java;

2. Note about the examples

The description of ISIS_DLL functions are accompanied by examples coded in Visual Basic and C++. These examples have the sole purpose of exemplifying the use of ISIS_DLL functions and are not intended to suggest any particular programming style or graphic user interface development methodology.

3. Installation

ISIS_DLL files are distributed via web. The following directory structure is recommended:

```
\ISISDLL\                   - ISIS_DLL root directory
        isis001.bas         - constants, user-defined types (Visual Basic)
        isis001.pas         - constants, record (Delphi)
        isis001.h           - constants, structures (C, C++)
        isisdll.h           - function prototypes (C,C++)
        ISIS32\             - ISIS_DLL for 32 bits applications
                isis32.dll  - ISIS_DLL library
                isis32.bas  - function declaration module (Visual Basic)
                isis32.pas  - function declaration unit (Delphi)
                isis32.lib  - function declaration library (C,C++ )
                isis32.jar   - function declaration library (Java)
```

For development (Windows OS), the Isis32.dll file should be copied to the Windows\System directory. It is also possible to set the environment variable PATH to the directory where the isis32.dll file is placed.

Chapter II. ISIS_DLL Overview

A Dynamic Link Library (DLL) is a special type of library used in the MS Windows Operating System[3]. It behaves like an executable module: when any of its functions is called by an application program, the DLL is loaded into memory and these may be further called simultaneously by other applications. This means that a single copy of the DLL may be shared by several applications, and that the DLL functions are external to the application modules using them. In principle, it is possible to upgrade a DLL without recompiling application programs.

ISIS_DLL is an ISIS Application Program Interface (API) for the MS Windows and Linux Operating Systems, i. e.; it contains functions specifically designed to handle ISIS entities. The main ISIS entities are: master file records, inverted file terms, formatting and search language specifications.

ISIS_DLL functions may be called from applications written in any language for which there is a compiler for the Windows operating system, such as Visual Basic, Delphi, C, C++, Java, PowerBuilder, etc.

Moreover, ISIS_DLL is designed in such a way that limitations regarding the number of master and inverted files opened as well as the number of records and terms loaded into memory at a given time depend exclusively on the computer resources available.

Since ISIS_DLL permits one to address specific problems as well as application interfaces not implemented in the standard ISIS system, it offers to programmers and system developers a great deal of flexibility for the development of (simple or complex) applications using a graphic user interface for processing ISIS data bases.

ISIS_DLL is designed for programmers. A prior knowledge of ISIS is highly recommended in order to use and better exploit its power. Programmers with previous knowledge of ISIS Pascal or the CISIS Interface will have no difficulties in using ISIS_DLL. ISIS Pascal programmers will probably prefer to code in Visual Basic and/or Pascal-like languages such as Delphi.

Conceptually, ISIS_DLL is based on three major programming entities: ISIS Application, ISIS Space and Shelf, as explained bellow.

[3] In Linux it is called Shared Library and has the extension *.so

1. ISIS Application

ISIS Application refers to any program, written in any language, that calls ISIS_DLL functions. In order to use ISIS_DLL, an application needs to call the IsisAppNew function and, before it terminates, the IsisAppDelete function.

The IsisAppNew function allocates and initializes memory areas for an application, displays the ISIS_DLL copyright notice and returns an application identification handle. IsisAppDelete terminates an ISIS Application and frees all memory allocated.

The following example shows calls to initialize and terminate an ISIS application:

```
A = IsisAppNew()
...

r = IsisAppDelete(A)
```

Since ISIS_DLL may be called by several applications at the same time, an application handle is the way the application identifies itself when calling the DLL.

ISIS Applications written in C and C++ should include the library *isis32.lib* and the header files *isis001.h* (ISIS constants and structures) and *isisdll.h* (functions prototypes) in their development project.

In a Visual Basic project, ISIS_DLL functions must be declared through a Declare statement before they can be referenced. The module *isis32.bas* includes Declare statements for all the available functions and *isis001.bas* contains ISIS constants and structures (user-defined types). ISIS applications in Visual Basic should include these files.

In a Delphi project, ISIS_DLL functions are defined in the unit module *isis32.pas*; constants and structures (records) are declared in the *isis001.pas* unit.

In a Java project, ISIS_DLL functions and ISIS constants and structures are contained in the isis32.jar file.

2. ISIS Space

ISIS Space refers to a set of ISIS resources, which correspond to a standard ISIS data base: one master file, one inverted file, data definition files, one format specification, one field select table, etc.

An ISIS application may open several ISIS Spaces.

The number or type of active resources in a ISIS Space is set by the application, i.e., a given ISIS Space may include both master and inverted files, while another ISIS Space may include only the inverted file, etc. Furthermore, the name of a resource, for example a master file name, may be changed at run time. Consequently, it is possible, for example, to open more than one standard ISIS data base in one application, or to have an application that manipulates one master file and several inverted files, or another application with two or more master files and one inverted file, etc.

A new ISIS Space is created by the IsisSpaNew function and deleted by IsisSpaDelete. IsisSpaNew returns a handle that is used as first parameter in all ISIS_DLL functions that perform operations on master file records and inverted file terms belonging to that particular ISIS space.

The following example shows the creation of two ISIS Spaces in an application identified by handle A:

```
H1 = IsisSpaNew(A)
H2 = IsisSpaNew(A)
```

These ISIS Spaces are destroyed by calling the IsisSpaDelete function:

```
r = IsisSpaDelete(H1)
r = IsisSpaDelete(H2)
```

Note that the function IsisAppDelete automatically deletes all the active spaces of the application. Thus, the IsisSpaDelete function is normally used when the deletion of an ISIS Space is required before terminating an ISIS Application.

In an ISIS Space, the identification of Master File, Inverted File and data base definition files is performed by specific functions: IsisSpaMf, IsisSpaIf, IsisSpaPft, IsisSpaFst, etc. For example, the following statements are used to specify the *CDS* master, inverted and format files in an ISIS Space identified by handle *H*:

```
filename = "c:\ISIS\data\cds"
r = IsisSpaMf(H, filename)
r = IsisSpaIf(H, filename)
filename = "@" + filename
r = IsisSpaPft(H, filename)
```

File name extensions (such as mst, xrf, cnt, etc.) are ignored.

Master and inverted files may be created or initialized by the functions IsisSpaMfCreate and IsisSpaIfCreate. The following example creates the data base *mydb*:

```
Dim A As Long           'Application handle
Dim H As Long           'Space handle
Dim r As Long           'return code
Dim s As String         'string work area
```

```
A = IsisAppNew()        'Starts an ISIS Application
H = IsisSpaNew(A)       'Creates a Space
s = "c:\bases\mydb"     'Data base path and name
r = IsisSpaMf(H, s)     'Master file name
r = IsisSpaIf(H, s)     'Inverted file name

r = IsisSpaMfCreate(H)  'Creates mydb.mst and myb.xrf files
r = IsisSpaIfCreate(H)  'Creates .cnt,.n01,.n02,.l01,.l02,.ifp files

r = IsisAppDelete(A)    'Deletes the application
```

The following example initializes the *CDS* inverted file:

```
filename = "c:\ISIS\data\cds"
r = IsisSpaIf(H, filename)
r = IsisSpaIfCreate(H)
```

3. Shelf and shelves

ISIS_DLL uses the concept of Shelf, initially introduced by the CISIS Interface, to designate a memory area, in an ISIS Space, where a master file record or an inverted file term is stored. ISIS Spaces are always initialized with two shelves, one to hold master file records and another to hold inverted file terms. At any given moment, a master file record shelf may only hold one record and an inverted file shelf may only hold one term.

The number of shelves can be modified by an application at run time. Shelves are identified by an index, starting at 0. Thus, in an ISIS Space with three shelves for master file records, these are identified by an index with values 0, 1 and 2.

The IsisSpaRecShelves and IsisSpaTrmShelves functions are used to redefine, in a given ISIS Space, the maximum number of shelves for master file records and inverted file terms respectively. The following example shows the initialization of a space capable to simultaneously hold 10 master file records in memory:

```
H = IsisSpaNew(A)
r = IsisSpaRecShelves(H, 10)
```

The default size of a record shelf is 30000 bytes, but it is possible to change it at run time. A Term shelf size is, by default, equal to 512 bytes. It may be zero, in which case the postings associated to the term are not loaded.

The IsisRecSize and IsisTrmSize functions permit to change the size of record and term shelves respectively. The following example allows a shelf to hold records with a maximum length of *16000 bytes*:

```
r = IsisRecSize(H, I, 16000)
```

4. IsisSpaDb Function

When an application handles data bases which have the complete set of CDS/ISIS files (master, inverted, format, etc.) as required by the standard CDS/ISIS 3.07, ISIS_DLL provides the function IsisSpaDb which includes all the required space initializations, i.e., IsisSpaMf, IsisSpaIf, IsisSpaPft, IsisSpaFst and IsisSpaStw. Function IsisSpaNew should be called before IsisSpaDb. The function IsisSpaDelete is used to close such a space.

The following example shows how to initialize and terminate an application using the *CDS* database:

```
Dim A, H As Long
Dim r As Long

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaDb(H,”c:\bases\cds\cds”)
...

r = IsisSpaDelete(H)
...

r = IsisAppDelete(A)
```

Chapter III. Master file processing: Data structure

An ISIS master file is a logical entity actually comprising two physical files: the master file with extension *.MST* and the cross-reference file with extension *.XRF*.

Although the actual data is stored only in the master file, the cross-reference file is an auxiliary file that provides a mechanism for the efficient processing of master file input and output operations.

The cross-reference file contains fixed length records, one for each master file record, with data which allow direct access to master file records and indicate the master file record status: active, logically or physically deleted, inversion pending or not.

Master file records are variable length records, each of them identified by a sequential number called master file number (Mfn). Master file records are stored sequentially and their position in the physical file is stored in the corresponding cross-reference file records. Thus, the address and status of master file record number *100* is stored in the *100th* record of the cross-reference file.

When a new master file record is created, it is stored at the end of the master file and a corresponding cross-reference record is created. When an existing master file record is updated, it is re-written at the end of the master file and its corresponding cross-reference file record is updated to reflect the new location and status. The modified master file record contains a link to its previous version in order to provide a mechanism for the inverted file updating process, i.e. the deletion of old inverted file term occurrences and the addition of new ones. After a record is inverted, the link to its old version is eliminated. The old record remains in the file (i.e. it occupies physical disk space) until a master file reorganization is done.

Master and cross-reference files consist of physical blocks of 512 bytes. Since a master file record length is variable a record may start at any position within a block. Hence, the physical address of a record is given by a block number and an offset within the block.

ISIS_DLL provides functions for master file input and output operations that automatically handle the physical storage of master and cross-reference files and the corresponding update mechanism.

ISIS_DLL also provides functions for import and export operations in ISO-2709 standard format.

An ISIS Application should call IsisSpaMf (or IsisSpaDb) to specify an ISIS Space master file name before calling any master file function.

1. Master file control record

Each master file has a record with Mfn *0* called Master File Control Record. It is always stored at the beginning of the file. It has a fixed length and it has no corresponding cross-reference record.

The control record has the following fixed data fields:

Field	Type	Description
ctlmfn	Long	master file number (always zero)
nxtmfn	Long	next master file number to be assigned
nxtmfb	Long	address of the next master file record: 512 bytes block number
nxtmfp	Long	address of the next master file record: offset within the block
mftype	Long	type of master file (always zero)
reccnt	Long	not used (always zero)
mfcxx1	Long	not used (always zero)
mfcxx2	Long	zero or number of applications with data entry lock
mfcxx3	Long	zero or 1 for exclusive write lock granted to one application

The above structure is identified by IsisRecControl. In order to have access to control record data, an application should define an IsisRecControl structure.

The function IsisRecControlMap moves the control record data fields to the IsisRecControl application area. The following example displays the last Mfn of the LILACS database:

```
Dim Control As IsisRecControl
Dim A, H As Long
Dim r As Long

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaMf(H, "c:\bases\lilacs\lilacs")
r = IsisRecControlMap(H, Control)

Print Control.nxtmfn - 1

r = IsisSpaDelete(H)
r = IsisAppDelete(A)
```

The IsisRecControl structure reflects the control record data at the moment IsisRecControlMap is called. Thus, when control record data is required in an environment where the master file is being updated, IsisRecControlMap should be called immediately before accessing the data in order to ensure that the latest situation is actually reflected.

IsisRecControlMap provides the application with read only access to the master file control record. ISIS_DLL automatically handles any required updating of the control record.

2. Master file records

A Master file record stores primary data, such as a bibliographic reference. All other ISIS files store secondary or dictionary data. The main characteristic of a master file record is to store variable length data, structured in data fields.

A record is loaded from a master file into an ISIS Space shelf by the function IsisRecRead (or IsisRecReadLock for update in concurrent updating environment). Records may be copied by the function IsisRecCopy from one shelf to another shelf, in the same or different spaces.

A record stored in a shelf may be written to the corresponding ISIS Space master file by the function IsisRecWrite (or IsisRecWriteLock or IsisRecWriteUnlock in a concurrent-updating environment). ISIS_DLL automatically handles any required updating of the cross-reference file and of the control record each time a record is written to the master file.

The following example shows how to copy master file CDS to CDS1:

```
Dim Control As IsisRecControl
Dim A, H, H1, Mfn As Long
Dim r As Long

A = IsisAppNew()
H = IsisSpaNew(A)                    ‘input db space
H1 = IsisSpaNew(A)                   ‘output db space

r = IsisSpaMf(H, "c:\bases\cds\cds")

r = IsisSpaMf(H1, "c:\bases\cds\cds1")
r = IsisSpaMfCreate(H1)

r = IsisRecControlMap(H, Control)
For Mfn = 1 To Control.nxtmfn - 1
    r = IsisRecRead(H, 0, Mfn)
    If r = ZERO Then
        r = IsisRecCopy(H, 0, H1, 0)  ‘copy record from H to H1
        r = IsisRecWrite(H1, 0)       ‘write record from H1
    End If
Next Mfn

r = IsisAppDelete(A)
```

2.1. Master file record structure

Master file records may be empty or contain one or more data fields. Thus, within the same master file, master records may have different lengths, may contain different data fields, which may have different lengths and may be stored in a different sequence inside each record.

Within a record, a data field is identified by an integer number (from 1 to 32767) called the Field Tag.

The ISIS master file record structure is similar to that of the ISO2709 record. According to this structure, a record contains three sections: leader, directory and data. Data fields are stored in the data section one after the other, without any separator. The directory section contains fixed length entries, which hold pointers to each data field stored in the data section. The leader has a fixed length and contains data on general characteristics of the record as a whole, such as Master file number (Mfn), record length, number of data fields, etc.

ISIS_DLL provides functions which permit read only access to the Leader and Directory sections of the master file record. It provides also several functions to access and modify the data section. In particular, ISIS_DLL provides support for two powerful command languages to handle data fields: the ISIS Field Update Language (to update data fields) and the ISIS Formatting Language (to extract and format data fields).

When a master file record is modified, ISIS_DLL automatically handles the update of all the required sections of the record. Thus, ISIS Applications do not have to care about the actual operations involved in the creation and/or update of the different sections of a master record.

2.1.1. Leader

The data structure of the leader is 28 bytes long:

Field	Type	Description
mfn	Long	master file record number
mfrl	Long	record length
mfbwb	Long	record's old version address: 512 bytes block number
mfbwp	Long	record's old version address: offset within the block
base	Long	data section starting position within the record
nvf	Long	number of fields or number of entries in the directory
status	Long	record status (0= record active, 1= record marked for deletion)

The above structure is defined by IsisRecLeader. In order to have access to leader data, an application should define an IsisRecLeader structure.

The function IsisRecLeaderMap moves the record leader data to the IsisRecLeader application area. The following example computes the minimum, maximum and average length of the CDS master file records:

```
Dim A, Mfn, H As Long
Dim r as Long
Dim Control As IsisRecControl
Dim Leader As IsisRecLeader
Dim min, max, total, avg As Single

A = IsisAppNew()
H = IsisSpaNew(A)

r = IsisSpaMf(H, "c:\bases\cds\cds")

min = 99999
max = 0
total = 0

r = IsisRecControlMap(H, Control)

For Mfn = 1 To Control.nxtmfn - 1
    r = IsisRecRead(H, 0, Mfn)
    If r = ZERO Then
        r = IsisRecLeaderMap(H, 0, Leader)
        If Leader.mfrl < min Then min = Leader.mfrl
        If Leader.mfrl > max Then max = Leader.mfrl
        avg = avg + Leader.mfrl
        total = total + 1
    End If
Next Mfn
avg = avg / total
Print min, max, avg

r = IsisSpaDelete(H)
r = IsisAppDelete(A)
```

Note that the minimum record length possible corresponds to an empty record, i.e. 18 bytes, which is the size of the leader section. An empty record is created either after a shelf initialization or after IsisRecNew or IsisRecNewLock or IsisRecDummy function calls.

2.1.2. Directory

The directory section has one entry for each field present in the record. Note that when a field, such as author's name, occurs more than once in a record, there will be an entry for each occurrence. The total number of directory entries is equal to the value of the nvf field in the Leader.

Each directory entry has the following structure (size = 12 bytes):

Field	Type	Description
tag	Long	data field tag
pos	Long	offset within record data section
len	Long	length of the data field

The above structure is defined by IsisRecDir. In order to have access to directory data, an application should define an array of IsisRecDir structures.

The function IsisRecDirMap moves the record directory to an application array of IsisRecDir structures. The following example shows the basic code to display the tag of fields present in a master file record:

```
Dim A, H, Mfn As Long
Dim Leader As IsisRecLeader
Dim Direct() As IsisRecDir
Dim r, k As Long

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaMf(H, "c:\bases\cds\cds")

Mfn = 1
r = IsisRecRead(H, 0, Mfn)

r = IsisRecLeaderMap(H, 0, Leader)

ReDim Direct(Leader.nvf)            ‘redimension Direct array

r = IsisRecDirMap(H, 0, 1, Leader.nvf, Direct(0))
For k = 1 To Leader.nvf
    Print Direct(k-1).tag
Next k

r = IsisSpaDelete(H)
r = IsisAppDelete(A)
```

Directory entries are stored in the same order in which the corresponding data fields were created. An empty record has no directory entries.

2.1.3. Data Fields

Data fields are stored in the record data section.

ISIS_DLL provides several functions to handle data fields: IsisRecField, IsisRecFieldN, IsisRecSubField(Ex), IsisRecDump, IsisRecFormat(Ex), IsisRecFieldUpdate, IsisRecUpdate.

The IsisRecField function provides access to individual data field occurrences. For example, the following statement retrieves the first occurrence of field *12* from a record, shelf 0 and moves it to *area12*:

```
r = IsisRecField(H, 0, 12, 1, area12, area12size)
```

A data field may contain one or more data elements called subfields. Each subfield is preceded by a two character delimiter consisting of a subfield indicator (the character ^), and a subfield identifier (one letter or one number). For example, a data field with three subfieds *a, b, c*, identifying respectively city, state and country, may have the following occurrence: *^aSao Paulo^bSP^cBrazil*. The function IsisRecSubField provides access to subfields. For example, the following statement retrieves subfield *a* of the first occurrence of field *26* from a record of the data base *CDS* and moves it to *area26a*:

```
r = IsisRecSubField (H, 0, 26, 1, ”a”, area26a, area26asize)
```

The first subfield of a field, which may, but need not contain an explicit subfield delimiter, may be referred to by *:

```
r = IsisRecSubField (H, 0, 26, 1, ”*”, area26a, area26asize)
```

ISIS_DLL provides the function IsisRecSubFieldEx to handle repeatable subfields, although they are not supported by standard ISIS format language.

2.1.4. ISIS Field Update Language specification

The ISIS Field Update Language is used to specify modifications of the contents of a record. The ISIS_DLL function IsisRecFieldUpdate executes one or more field update commands.

The following ISIS Field Update commands are available:

d.	Makes the record logically deleted
d*	Deletes all present fields
dtt	Deletes all occurrences of field tt
dtt/occ	Deletes occurrence occ of field tt
att#str#	Adds string str as a new occurrence of field tt
htt n str_n	Adds string str_n, n bytes long, as a new occurrence of field tt
=n	Assigns n to MFN
s	Sorts the directory entries by tag value

Some examples of field update commands are given bellow:

a. add fields 10 and 12, using command a and the symbol # as data separator:

```
u = "a10#Magalhaes, Elisabeth#a12#Project analysis and evaluation#"
r = IsisRecFieldUpdate(H, 0, u)
```

b. delete the second occurrence of field 87:

```
u = "d87/2"
r = IsisRecFieldUpdate(H, 0, u)
```

The following example shows how to add the string *CDS* into field *1* of all records of *CDS* database:

```
Dim A, H, Mfn As Long
Dim x As String
Dim r As Long
Dim Control As IsisRecControl

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaMf(H, "c:\bases\cds\cds")

r = IsisRecControlMap(H, Control)

For Mfn = 1 To Control.nxtmfn - 1
    r = IsisRecRead(H, 0, Mfn)
    If r = ZERO Then
        x = "a1!CDS!"
        r = IsisRecFieldUpdate(H, 0, x)
        r = IsisRecWrite(H, 0)
    End If
Next Mfn

r = IsisSpaDelete(H)
r = IsisAppDelete(A)
```

2.1.5. IsisRecDump and IsisRecUpdate

IsisRecDump and IsisRecUpdate functions execute opposite actions.

IsisRecDump moves all the fields of a record to an application area, according to the following format:

- each field is separated from the next one by Carriage Return and Line Feed characters;
- each field is preceded and followed by special begin and end delimiters, which identify the field tag;
- the begin delimiter has the format *<tag>* and the end delimiter has the format *</tag>*.

The following is an example of IsisRecDump output when applied to record 1 of the *CDS* database:

```
<44>Methodology of plant eco-physiology: proceedings of the
Montpellier Symposium</44>
<50>Incl. bibl.</50>
<69>Paper on: <plant physiology><plant transpiration><measurement
and instruments></69>
<24>Techniques for the measurement of transpiration of individual
plants</24>
<26>^aParis^bUnesco^c-1965</26>
<30>^ap. 211-224^billus.</30>
<70>Magalhaes, A.C.</70>
<70>Franco, C.M.</70>
```

IsisRecUpdate processes an application program string, according to the above format and moves it to the record stored on the corresponding shelf. All previous record contents are destroyed. Thus, for example, the following sequence of operations does not alter the contents of the record:

```
r = IsisRecRead(H, 0, Mfn)
r = IsisRecDump(H, 0, Area, AreaSize)
r = IsisRecUpdate(H, 0, Area)
r = IsisRecWrite(H, 0)
```

The combination of IsisRecDump and IsisRecUpdate provides a powerful mechanism to update a master file record.

The default begin and end tag delimiters are < and >. The function IsisSpaRecDelim can be used to change them.

The following example shows how to change the delimiters from the default to *{* as begin delimiter and *}* as end delimiter:

```
delim1 = “{“
delim2 = “}”
r = IsisSpaRecDelim(H, delim1, delim2)
```

2.1.6. Creating New Records

ISIS_DLL provides two functions to create new records: IsisRecNew (or IsisRecNewLock for concurrent operations) and IsisRecDummy.

IsisRecNew creates an empty record in both the master file and the Isis Space. IsisRecDummy creates an empty record only in the Isis Space.

In both cases, data may be added to the record in the shelf via IsisRecFieldUpdate or IsisRecUpdate.

IsisRecWrite (or IsisRecWriteLock or IsisRecWriteUnlock for concurrent operations) is used to actually write the record in the master file.

2.1.7. Execution of ISIS Format Language specification

The ISIS Formatting Language[4] is used to extract all or a subset of fields and/or subfields from a record and format them. The ISIS_DLL function IsisRecFormat executes a sequence of formatting commands. Examples of formatting commands are:

a. format field 70 and store the result in Areaf:

```
f = "v70+|; |"
r = IsisSpaPft(H, f)
r = IsisRecFormat(H, 0, Areaf, AreafSize)
```

b. print the Mfn of the record if fields 44 and 24 are present:

```
f = " if p(v44) and p(v24) then mfn fi "
r = IsisSpaPft(H, f)
r = IsisRecFormat(H, 0, Areaf, AreafSize)
If Areaf <> "" Then Print Left$(Areaf, InStr(Areaf, Chr(0)) - 1)
```

The following example shows how to display *CDS* records according to a format specification stored in the file *CDS.PFT*:

```
Dim A, H, Mfn As Long
Dim r, rf As Long
Dim f As String
Dim Areaf As String * 8000

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaDb(H, “c:\bases\cds\cds”)

r = 0
Mfn = 1
While r <> ERR_RECEOF
   r = IsisRecRead(H, 0, Mfn)
   If r = 0 Then
      Areaf = ""
      rf = IsisRecFormat(H, 0, Areaf, AreafSize)
      Print Areaf
   End If
   Mfn = Mfn + 1
Wend

r = IsisAppDelete(A)
```

[4] A full description of the ISIS Formatting Language is given in the *CDS/ISIS Reference Manual*

2.2. Importing and Exporting records from/to ISO-2709 file.

ISIS_DLL provides the IsisRecIsoRead function to import records from an ISO-2709 file and IsisRecIsoWrite to export records. The following example shows how to import a data base from a ISO-2709 file and how to export the same database to another ISO-2709 file.

```
Dim A, H, Mfn As Long
Dim r As Long

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaMf(H, “c:\temp\cds”)
r = IsisSpaMfCreate(H)
r = IsisSpaIsoIn(H, “c:\isis\data\cds\cds.iso”)
r = IsisSpaIsoOut(H, “newcds.iso)

r = 0
Mfn = 1
While r <> ERR_RECEOF
   r = IsisRecIsoRead(H, 0, Mfn) ‘Import a record.
   r = IsisRecWrite(H, 0)        ‘It is necessary to store the record.
    r = IsisRecIsoWrite(H, 0)     ‘Export a record.
   Mfn = Mfn + 1
Wend

r = IsisAppDelete(A)
```

2.3 Compressing and/or encrypting records: The global change of patterns.

ISIS_DLL provides a mechanism that changes the contents of each record read from a database, by changing sequences of characters of their fields by another ones according a conversion table. This conversion process is called gizmo mechanism and it is activated by calling the ISIS_DLL’s function IsisSpaGf.

The gizmo mechanism is supposed to be used to compress the fields of the records by changing sequences of characters that repeat too much by symbols or by shorter sequences. It can be also used to encrypt the records by changing the characters of their fields by others, and also to change accented letter by others (like café by cafe).

To use the gizmo mechanism, it is necessary to create a database where each record has 2 fields, the first one, tag 1, indicates the sequence of chars to be changed and the second field, tag 2, indicates the sequence of chars that will substitute the first one.

Let’s suppose we have the following gizmo database:

```
mfn=     1
  1  «<»
  2  «/»

mfn=     2
  1  «>»
  2  «/»

mfn=     3
  1  «plants»
  2  «PLANTS»
```

and a typical database record:

```
<24>Techniques for the measurement of transpiration of individual
plants</24>
<26>^aParis^bUnesco^c-1965</26>
<30>^ap. 211-224^billus.</30>
<70>Magalhaes, A.C.</70>
<70>Franco, C.M.</70>
<44>Methodology of plant eco-physiology: proceedings of the
Montpellier Symposium</44>
<50>Incl. bibl.</50>
<69>Paper on: <plant physiology><plant transpiration><measurement
and instruments></69>
```

if we the gizmo is applied to this record, the following record will be obtained:

```
<24>Techniques for the measurement of transpiration of individual
PLANTS</24>
<26>^aParis^bUnesco^c-1965</26>
<30>^ap. 211-224^billus.</30>
<70>Magalhaes, A.C.</70>
<70>Franco, C.M.</70>
<44>Methodology of plant eco-physiology: proceedings of the
Montpellier Symposium</44>
<50>Incl. bibl.</50>
<69>Paper on: /plant physiology//plant transpiration//measurement
and instruments/</69>
```

If it is necessary to revert the process, two conversion databases are needed, one for the compression/encryption process and another for the decompression/decryption.

The following example shows how to create a modified database and how to create the original database from the modified one.

```
Dim A, H, H1, Mfn As Long
Dim r As Long

A = IsisAppNew()
```

```
H = IsisSpaNew(A)
H1 = IsisSpaNew(A)

r = IsisSpaMf(H, “original”)
r = IsisSpaMf(H1, “compressed”)
r = IsisSpaMfCreate(H1)
r = IsisSpaGf(H, “orig2comp”)

r = 0
Mfn = 1
While r <> ERR_RECEOF
   r = IsisRecRead(H, 0, Mfn)
   If r == ZERO Then
      r = IsisRecCopy(H, 0, H1, 0) `Copy records from different dbs.
      r = IsisRecWrite(H1, 0)      ‘Store the record.
   End If
   Mfn = Mfn + 1
Wend

r = IsisSpaMf(H, “original2”)
r = IsisSpaMfCreate(H)
r = IsisSpaGf(H1, “comp2orig”)

r = 0
Mfn = 1
While r <> ERR_RECEOF
   r = IsisRecRead(H1, 0, Mfn)
   If r == ZERO Then
      r = IsisRecCopy(H1, 0, H, 0) `Copy records from different dbs.
      r = IsisRecWrite(H, 0)      ‘Store the record.
   End If
   Mfn = Mfn + 1
Wend


r = IsisAppDelete(A)
```

Chapter IV. Inverted File Updating / Loading

The ISIS Inverted file is a set of six physical files which permit retrieval of master file records using a term or a combination of terms (see Chapter V for a detailed explanation of inverted file access). Each term appears only once in the inverted file, associated with a list of postings (Mfn, field identifier as defined in the FST, occurrence and position). Therefore, when a master file record is updated, it may require the updating of an associated inverted file in order to ensure that terms and/or postings represent the new record contents. The process of updating the inverted file from a master file record is also called record inversion. When the inverted file is cleaned and all master file records are inverted, the update process is called inverted file loading.

As described earlier, when a master file record is rewritten to the master file, ISIS_DLL appends the record at the end of the master file, creates a link to the record's old version corresponding to the last inverted file update and activates an inverted file pending indicator in the cross-reference record. Thus, if a master file is updated three times without updating the inverted file, the current record points to the first version of the record and the other intermediary versions are not accessible and therefore waste disk space. An inverted file update is done in two stages: first, all terms postings of the of the old version are deleted and second, new terms and new postings of the current record are added to the inverted file. Note that an inverted file update does not delete terms but only postings; therefore, after an inverted file update, the dictionary may contain terms with no postings.

Terms and postings are extracted from a master file record according to a Field Select Table (FST) specification. A FST contains one or more lines. Each line contains three fields: a field identifier, an indexing (or term extraction) technique and a format specification. The term extraction process works in the following way: for each FST line, the format is executed and one or more output lines are stored in a memory area; the indexing technique is then applied to each line to generate one or more terms (each term occurrence is followed by the corresponding posting, i.e., the Mfn, the FST field identifier, the occurrence and the ordinal position of the term). After all FST lines are processed, the inverted file is updated in the following way for each term/posting extracted: a term which does not exist in the inverted file, is added together with its posting; a term which already exists has its posting list updated with the new posting.

The function IsisSpaFst is used to specify the Isis Space FST.

ISIS_DLL provides the function IsisRecIfUpdate to perform the inversion of a master file record. ISIS_DLL automatically handles all the processing related to

inverted file updating, including deletion of postings and addition of new terms and postings, the elimination of the record link to its old version and of the cross-reference inverted file pending indicator.

The following statement updates the inverted file from a master file record with master file number *Mfn,* defined in ISIS Space *H*:

```
r = IsisRecIfUpdate(H, Mfn)
```

The function assumes that an ISIS Space inverted file name was previously identified by a call to the function IsisSpaIf.

It is important to note that the function IsisRecIfUpdate does not require the presence of a link to an old record version and/or the presence of the cross reference inverted file pending indicator. This permits the inversion of newly created records as well as several inversions of the same record.

When an application has a master file with more than one associated inverted files, master file record inversion may be done in two ways. By using several ISIS spaces, one for each inverted file, or, alternatively, by using only one ISIS Space and changing the inverted file before calling IsisRecIfUpdate. Note that, in the first case, a copy of the master record is required in all ISIS Spaces.

The inverted file loading is done by the IsisRecLnk function which generate links from a range of mfs, IsisLnkSort which sorts the keys of the generated links and finally by the IsisLnkIfLoad function which empties the inverted file and updates it with the generated postings.

The next example shows how to update the inverted file with the contents of a new record:

```
Dim A, H As Long
Dim r As Long
Dim mfn As Long

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaMf(H, "c:\bases\cds\cds")
r = IsisSpaIf(H, "c:\bases\cds\cds")
r = IsisSpaFst(H, “24 4 MHU,V24”)  ‘r = IsisSpaFst(H, "@c:\bases\cds\cds")

mfn = IsisRecNew(H, 0)
r = IsisRecFieldUpdate(H, 0, “a24#assurances sociales#)
r = IsisRecWrite(H, 0)
r = IsisRecIfUpdate(H, mfn)
```

The following example shows how to load an inverted file from a master:

```
Dim A, H As Long
Dim r As Long
```

```
A = IsisAppNew()
H = IsisSpaNew(A)

r = IsisSpaMf(H, DBNAME);
r = IsisSpaIf(H, DBNAME);

r = IsisSpaFst(H, "@biblo.fst");  ‘Field select table
r = IsisSpaStw(H, "@" + DBNAME);  ‘Stop words
r = IsisAppUcTab(A, "@c:\\isis\\data\\cds\\isisuc");

r = IsisRecLnk (H, 1, 9999999); ‘link generation
r = IsisLnkSort(H);             ‘sort of the links
r = IsisLnkIfLoad(H);           ‘load process

r = IsisAppDelete(A);
```

The following section describes master file record inversion in a concurrent (multi-user) environment.

1. Concurrent master and inverted files updating

ISIS_DLL provides functions for updating master and inverted files in a multi-user environment. Under the Windows operating system, either in a stand-alone mode or in a network, one or more ISIS Applications may coexist with other instances of the ISIS Standard Systems. In these situations, ISIS Applications which execute input and output operations over the same master and/or inverted files, should follow the standard ISIS locking mechanism as provided by the ISIS_DLL update functions.

The execution of a function will not necessarily succeed, for example, ISIS_DLL will not load a master file record for updating if it is already being updated by another application. Therefore, the application program should always check the return code of concurrent update functions in order to comply with the ISIS standard locking mechanism and therefore avoid incorrect results.

The IsisRecReadLock function should be used to load a master file record for updating into a shelf of an ISIS Space with concurrent updating rights. The call may succeed or not. It will not succeed if the master file record is already locked by another process or if the master file is locked by another application (for example, a master file is locked by ISIS_DLL during the inversion of a master file record.). It may also be locked by a standard ISIS application while performing inverted file updating and loading, importing ISO records, etc. When IsisRecReadLock succeeds, ISIS_DLL locks the master file record. This is done by rewriting the master file with a negative length (leader field mfrl). The lock will remain until the application unlocks the record.

The following example prints the return code of an IsisRecReadLock call:

```
r = IsisRecReadLock(H, 0, Mfn)
Select Case r
   Case ZERO
       Print Str$(Mfn) + " loaded!"
   Case ERR_RECLOCKED
       Print Str$(Mfn) + " not loaded: locked by another application"
   Case ERR_RECLOGIDEL
       Print Str$(Mfn) + " loaded: logically deleted"
   Case ERR_PHYSDEL
       Print Str$(Mfn) + " not loaded: physically deleted"
   Case ERR_RECEOF
       Print Str$(Mfn) + " not loaded: eof"
End Select
```

The IsisRecWriteLock or IsisRecWriteUnlock should be used to write a master file record by respectively maintaining the record locked or unlocking it after it has been written. The call may succeed or not! It will not succeed if the master file was locked meanwhile by another application.

The following example prints the result of an IsisRecWriteLock call, which also applies to IsisRecWriteUnlock:

```
Select Case r
   Case ZERO
      r1 = IsisRecWriteLock(H, 0)
      if r1 = ZERO then Print “Update succeed!”
   Case ERR_DBDELOCK
      Print “Update failed: Master file locked by another application”
.........
End Select
```

The function IsisRecUnlock permits to unlock one record without rewriting it via IsisRecWriteLock or IsisRecWriteUnlock.

A master file record inversion is performed by the IsisRecIfUpdate function. ISIS_DLL automatically handles all necessary locking when performing inverted file updating and inverted file loading. However, the call may succeed or not! Since inversion/loading requires a lock of the master file and of the inverted file, a call to IsisRecIfUpdate/IsisLnkIfLoad will fail if either the master or inverted files are already locked by another application.

The following example prints the return code of an IsisRecIfUpdate call:

```
r = IsisRecIfUpdate(H, Mfn)
if r = ERR_DBDELOCK then Print “Inversion failed: master file
locked by another application”
```

Chapter V. Inverted File Processing: retrieval

As mentioned above, “inverted file” is a logical name that refers actually to 6 physical files, which provide a mechanism for fast term input and output. Inverted file data are commonly generated from master file records via two processes: inverted file load and inverted file update (both may be done by either standard ISIS/CISIS utilities or the ISIS_DLL IsisRecIfUpdate/IsisLnkIfLoad functions)

Terms are divided into two groups, according to their length: group 1 includes terms up to 10 characters long and group 2 terms with a length of 11 to 30 characters. For each group, ISIS implements a B*tree data storage and access structure. Each B*tree corresponds to two physical files: one for nodes (extension .N01 for group 1 and .N02 for group 2) and another for leafs (extension .L01 for group 1 and .L02 for group 2). The ISIS term dictionary is obtained by merging the two leaf files. There is an inverted control file (extension .CNT) with two records containing control data for each B*tree. The sixth file (extension .IFP) contains term postings for both groups, i.e., it contains a list of postings for each dictionary term; each posting identifies the source of the corresponding term occurrence in the database (Mfn, identifier, occurrence and position). Therefore, each inverted file input or output operation involves the processing of one control record, one node file, one leaf file and the posting file.

ISIS_DLL provides functions for term input and output which automatically handle the processing and maintenance of all the physical files of the inverted files. In addition, ISIS_DLL provides a function for the processing of the ISIS Search Language.

1. Inverted file terms

A term is loaded from an inverted file into an ISIS Space term shelf by the function IsisTrmReadMap. This function is independent from term size: it always assumes a maximum length of 30 characters. ISIS_DLL decides which B*Tree has to be processed in each case. In addition to the term itself, the read function may also load the corresponding first postings block, unless the term shelf size is set to 0. The following example shows how to load the term *plant* into shelf *0* of the ISIS Space identified by handle *H*:

```
Dim trmread As IsisTrmRead

trmread.key = "Plant"
r = IsisTrmReadMap(H, 0, trmread)
```

Note that if term does not exist, IsisTrmReadMap returns a *not found* code and, unless the end of the dictionary is reached, loads the next (alphabetically higher) term into the shelf.

After an IsisTrmReadMap is issued, it is possible to read the inverted file sequentially. The sequential read is performed by IsisTrmReadNext. The following example shows how to print all terms of *CDS* inverted file:

```
Dim A, H As Long
Dim trmread As IsisTrmRead
Dim r As Long

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaIf(H, "c:\bases\cds\cds")

trmread.key = ""
r = IsisTrmReadMap(H, 0, trmread)

While r <> ERR_TRMEOF
    Print trmread.key
    r = IsisTrmReadNext(H, 0, trmread)
Wend

r = IsisAppDelete(A)
```

IsisTrmReadMap returns the number of postings whenever the requested term is found. If the term is not found and the end of the dictionary is reached, it returns ERR_TRMEOF, otherwise it returns ERR_TRMNEXT and moves into the term area the next term found. In the latter case, in order to obtain the number of postings of the returned term, it is necessary to issue an IsisTrmReadMap again.

IsisTrmReadNext returns the number of postings of the next term and moves the term into the application's term area parameter.

The following example prints the total number of postings for each term in the *CDS* inverted file:

```
Dim A, H As Long
Dim r As Long
Dim trmread As IsisTrmRead

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaIf(H, "c:\bases\cds\cds")

trmread.key = ""
r = IsisTrmReadMap(H, 0, trmread)

While r <> ERR_TRMEOF
    Print r, trmread.key
    r = IsisTrmReadNext(H, 0, trmread)
Wend
```

```
r = IsisAppDelete(A)
```

1.1. Term postings

An inverted file term may have no posting, or have one or more postings. Each posting has a data structure that identifies the source of the term occurrence in the database.

Note that if IsisTrmSize was issued with value 0, no postings are made available!

A posting structure has 20 bytes:

Field	Type	Description
posting	Long	posting sequential number
mfn	Long	master file record mfn that originated the term occurrence
tag	Long	field identifier, as defined in the FST table
occ	Long	field occurrence
cnt	Long	position within the field

The above structure is defined by IsisTrmPosting. In order to have access to posting data, an application should define an IsisTrmPosting structure.

The function IsisTrmPostingMap permits the retrieval of whole or a subset of postings to an application IsisTrmPosting array. The following example displays all terms and corresponding postings of the *CDS* inverted file:

```
Dim A, H As Long
Dim r, rp, i As Long
Dim TrmRead As IsisTrmRead
Dim Posting() As IsisTrmPosting

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaIf(H, "c:\bases\cds\cds")

TrmRead.key = ""
r = IsisTrmReadMap(H, 0, TrmRead)  'r = ERR_TRMNEXT is returned
r = IsisTrmReadMap(H, 0, term) 'read again to retrieve the postings

While r <> ERR_TRMEOF
    Print r, TrmRead.key

    ReDim Posting(r)

    rp = IsisTrmPostingMap(H, 0, 1, r, Posting(0))
    For i = 1 To r
       Print Posting.Posting,Posting.mfn,Posting.tag,Posting.occ,Posting.cnt
    Next i
  r = IsisTrmReadNext(H, 0, TrmRead)
Wend
```

```
r = IsisAppDelete(A)
```

When only the mfn posting component is desired, ISIS_DLL provides the function IsisTrmMfnMap, which permits the retrieval of whole or a subset of the mfns associated to a inverted file term. In this case, postings with duplicated mfns are ignored. Mfns are moved to an array of IsisTrmMfn structure. The following example dumps all CDS records retrieved by term “water”:

```
Dim A As Long                       'application handle
Dim H As Long                       'space handle
Dim r As Long                       'return code
Dim trmread as IsisTrmRead          'IsisTrmRead structure
Dim area As String * 2000           'maximum record
Dim s As String                     'work string
Dim from_Posting As Long            'first posting
Dim to_Posting As Long              'to posting
Dim i As Long                       'work long

Dim pIsisTrmMfn() As IsisTrmMfn     'array of mfns

  A = IsisAppNew()                  'creates new Isis application
  H = IsisSpaNew(A)                 'create a new Isis Space

  s = "cds"                        'cds path
  r = IsisSpaIf(H, s)               'space inverted file path
  r = IsisSpaMf(H, s)               'space master file path

  trmread.key = "water"             'term is water
  r = IsisTrmReadMap(H, 0, trmread) 'read term s into shelf 0
  If r > 0 Then                     'found term? if yes, r= number of postings
      from_Posting = 1              'from mfn posting 1
      to_Posting = r                'to mfn posting r
      ReDim pIsisTrmMfn(r)          'redim array of mfns
      r = IsisTrmMfnMap(H, 0, from_Posting, to_Posting, pIsisTrmMfn(0))
      For i = 0 To r - 1            'loop all mfns
          r = IsisRecRead(H, 0, pIsisTrmMfn(i).mfn)      'read next mfn
          r = IsisRecDump(H, 0, area, 2000)              'dump records
          Print Left$(area, r)      'display area
      Next i

  Else

      MsgBox "Term: " + s + " not found!"

  End If

  r = IsisAppDelete(A)

End Sub
```

Note that the array of mfns are resized to its maximum upper bound value. When postings have repeated mfns, the number of actual mfns moved is less than the number of postings; the remaining array cells are set to zero. The IsisTrmMfnMap return code indicates the actual number of mfns moved to the application area.

2. ISIS Search Language

The ISIS Search Language permits the specification of a search expression by combining terms with Boolean operators. The search engine retrieves the terms from an inverted file and executes the Boolean operations over the term postings.

Terms are strings of up to 30 characters; if an expression contains terms longer than 30 characters, they are automatically truncated. Terms may be represented by a root followed by the truncation indicator $; for example, *plant$* will retrieve the postings of all terms beginning with the string *plant*. Each search expression is identified by a number (#1, #2, #3, etc.). Search expression number may be used as terms.

Given two terms A and B, the ISIS Search Language accepts the following Boolean operators (listed from lowest to highest priority in expression evaluation):

Operation	Operator	Example	Posting element considered in the Boolean operation
Or	+	A+B	Mfn: result is the sum of all A and B postings
And	*	A*B	Mfn: result is A and B postings having same Mfn
And Not	^	A^B	Mfn: result is A postings with Mfn not present in B postings
Group	(G)	A(G)B	Mfn and Id: result is A and B postings having same Mfn and Id
Field	(F)	A(F)B	Mfn, Id and Occ: result is A and B postings having same Mfn, Id and Occ
Adjacency	.	A . B	Mfn, Id, Occ and Pos: result is A and B postings having same Mfn, Id, Occ and a difference between the corresponding Pos not higher than the number of periods
Adjacency	$	A $ B	Mfn, Id, Occ and Pos: result is A and B postings having same Mfn, Id, Occ and a difference between the corresponding Pos exactly equal to the number of dollar signs

In expressions with more than 2 terms, parenthesis can be used to change evaluation priority. ISIS_DLL does not support search expressions containing ANY keys.

As the search engine operates over term postings, the result of a search consists also of a list of postings. The results are stored in a temporary file called Search Log. An ISIS Application may have one or more Search Logs identified by numbers, starting at 0. So, we can have search logs 0, 1, 2, etc. In each log, searches have their own independent number, starting at 1. The Search Log files

are created: [Windows] into the application directory or into a directory set by the environment variable “TEMP”. [Linux] into the standard temporary directory “/tmp”.

A search expression always applies to a specific inverted file, identified by the ISIS Space handle. When an application permits searching on more than one inverted file, it must submit each search expression (possibly modified as required) for each inverted file. If several inverted files correspond to one master file, it is possible to combine in a Boolean expression, search numbers of searches executed on different inverted files.

ISIS_DLL provides basic functions to handle ISIS Search Language expressions and search log files.

The function IsisSrcSearch executes a search expression over an ISIS Space Inverted file, stores the result in the search log file and moves basic resulting data of the expression executed into an application area called search header structure.

A search header structure has the following data:

Field	Type	Description
Number	Long	Search number
Recs	Long	Total number of records
Dbname	String	Data base name, Size=64
Expr	String	Search expression, Size=512

The above data structure is defined by IsisSrcHeader. In order to execute a search expression, an application should define an IsisSrcHeader structure.

The following example prints the total number of records, stored in Search Log *0*, resulting from a search executed on an ISIS Space with handle *H*:

```
Dim SearchArea As IsisSrcHeader

r = IsisSrcSearch(H, 0, "plant * water", SearchArea)
Print SearchArea.recs
```

Note that in the search header structure, the element recs (total number of records) indicates the actual number of different mfns retrieved.

ISIS_DLL provides the function IsisSrcMfnMap, which permits the retrieval of whole or a subset of mfns retrieved. Mfns are moved to an array of IsisSrcMfn structure. The following example dumps all CDS mfn records retrieved by search expression “plant * water”:

```
Dim A, H As Long
Dim query As String
Dim r, i As Long
```

```
Dim SearchStru As IsisSrcHeader
Dim SrcMfn() As IsisSrcMfn

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaIf(H, "c:\bases\cds\cds")

query = "plant * water"
r = IsisSrcSearch(H, 0, query, SearchStru)
If r > 0 Then
 Redim SrcMfn (SearchStru.recs)
 r = IsisSrcMfnMap (A, 0, 0, 1, SearchStru.recs, SrcMfn(0))
  If r > 0 Then
   For i = 1 To SearchStru.recs
    Print SrcMfn(i-1).mfn
   Next i
  End If
End If

r = IsisAppDelete(A)
```

One application may have one or more temporally search log files (up to 200). Temporally search log files are created by the function IsisSrcSearch and they are deleted when the application terminates. The function IsisSrcLogFileFlush permits to delete the contents of an entire Search Log at run-time. The following statement shows how to clear search log number *0*:

```
r = IsisSrcLogFlush(H, 0)
```

The function IsisSrcLogFileSave is used to save permanently the contents of a search log file in a given moment. The saved file can then be reused by IsisSrcLogFileUse, allowing the use of search log files across different applications or different sections of the same application. The following example shows a call to IsisSrcLogFileSave:

```
s = “c:\bases\cds\cds.log”            ‘search log file
r = IsisSrcLogFileSave (H, 0, s)‘save log file 0 into s
```

The following example shows a call to IsisSrcLogFileUse:

```
s = “c:\bases\cds\cds.log”            ‘search log file
r = IsisSrcLogFileUse (H, 0, s)       ‘use s as search log file 0
```

3. Advanced Search

During the search, ISIS_DLL uses the inverted file defined in the Isis Space via IsisSpaIf function.
Using the default search mechanism, it is only possible to use one inverted file when a Boolean expression is executed, but ISIS_DLL provides an extended

search mechanism that allows a search expression indicates how many inverted files and which ones should be used.
The extended search can be used through IsisSrcSearchEx function, which has as one of its parameters a “search conversion table” of the form:

^p<prefix> ^y<inverted> ^u<use>

associating prefixes with inverted files as described bellow.

Let us suppose a database with records of authors and titles of books. It can have three inverted files of this database: one having only authors (Authors), other with only titles (Titles) and another having both authors and titles (AuthorsTitles). The search table could be the following (one association per line):

```
^p*       ^yAuthorsTitles
^pAUT  ^yAuthors
^pTIT   ^yTitles
```

This table says if we don’t use any prefix, the search should use the ´AuthorsTitle´ inverted file (the default one), if the AUT prefix is specified then the ´Authors´ inverted file should be used and if TIT prefix is specified then the ´Titles´ inverted file should be used.

Some examples of Boolean expressions could be:

```
“Luziadas * Camoes”                 // AuthorsTitle used.
“AUT Camoes”                        // Authors used.
“TIT Luziadas”                      // Title used.
“AUT Shakespeare + TIT Luziadas + Camoes” // Authors, Title and AuthorsTitle used.
“[AUT] (Shakespeare + Camoes) * TI Luziadas” // Authors and Title used.
```

Let us suppose now, the database uses only one inverted file (ALL) for title and author, but we add, via fst, a ´TI=´ prefix to the title’s keys and ´AU=´ to the author’s keys to distinguish each other. The search table then, could be the following:

```
^p*       ^yALL
^pAUT ^yALL ^uAU
^pTIT  ^yALL ^uTI
```

Here, the first line says that if no prefix is used, look up the key the way it was written in ALL inverted file.
The second line says that if a prefix AUT is used, look up the key in ALL inverted file but before that, add a ‘AU=’ prefix to the key.
And the third line says to add a ‘TI=’ prefix to the key if a prefix TIT is used, and then look it up in the ALL inverted file.

Some examples of Boolean expressions could be:

“Luziadas * Camoes” // nothing will be found.
“AUT Camoes” // Ok. The real key is AU=CAMOES.
“TIT Luziadas” // Ok. The real key is TI=LUZIADAS.
“[AUT] (Shakespeare + Camoes) * TI Luziadas” // OK.

If the Boolean expression “TIT Camoes” were used, nothing would be found because the real key is ‘AU=Camoes’ and not ‘TIT=Camoes’, and if the Boolean expression “AUT Shakespeare + TIT Luziadas + Camoes” were used, nothing related with ´Camoes´ would be found because in the ALL inverted file the key would be ‘AU=Camoes’ and not ‘Camoes’.

The following example dumps all BIBLO mfn records retrieved by the search expression “[AUT] (Shakespeare + Camoes) * TI Luziadas” and search conversion table at bibtable.tab:

```
Dim A, H As Long
Dim query As String
Dim r, i As Long
Dim SearchStru As IsisSrcHeader
Dim SrcMfn() As IsisSrcMfn

A = IsisAppNew()
H = IsisSpaNew(A)

query = "[AUT] (Shakespeare + Camoes) * TI Luziadas"
r = IsisSrcSearchEx(H, 0, @bibtable, query, SearchStru)
If r > 0 Then
 Redim SrcMfn (SearchStru.recs)
 r = IsisSrcMfnMap (A, 0, 0, 1, SearchStru.recs, SrcMfn(0))
  If r > 0 Then
   For i = 1 To SearchStru.recs
    Print SrcMfn(i-1).mfn
   Next i
  End If
End If

r = IsisAppDelete(A)
```

4. Regular Expression Search

The ISIS_DLL allows the use of regular expressions to do a search into the master file records. This kind of search does not use the inverted file, the velocity of the search is decreased, but it allows a kind of search that is not possible with the traditional Boolean expression.

A regular expression is one or more occurrences of one or more characters optionally enclosed in quotes. The following symbols are treated specially:

^	Start of line
$	end of line

.	any character
\	Quote next character
*	Match zero or more
+	Match one or more
[aeiou0-9]	Match a, e, I, o, u, and 0 through 9
[^aeiou0-9]	Match anything but a, e, i, o, u, and 0 through 9

Given a regular expression and a range of mfns where the expression will be applied, the IsisSrcRegExpMap function will return an array of structures IsisSrcRegExp with the following fields:

mfn – the mfn of a record where the regular expression has matched.
tag – the tag of a record where the regular expression has matched.
expr – the sentence which matched with the regular expression.

The following example shows the first 100 words that contain the word “ion” from mfn 1 to 10.

```
Dim A, H As Long
Dim regexp As String
Dim r, i As Long
Dim RegExpStru(100) As IsisSrcRegExp

A = IsisAppNew()
H = IsisSpaNew(A)
r = IsisSpaIf(H, "c:\bases\cds\cds")

regexp = "ion"
r = IsisSrcRegExpMap(H, regexp, 1, 10, RegExpStru(0), 100);
If r > 0 Then
    For i = 1 To r
     Print RegExpStru.expr + NewLine
    Next i
End If

r = IsisAppDelete(A)
```

Chapter VI. ISIS_DLL Functions[5]

IsisAppAcTab

#include<isisdll.h>

Long
FAR PASCAL

IsisAppAcTab*(long apphandle,*
*char *actab)*

The **IsisAppAcTab** function specifies the Isis alphabetic character table used in word indexing technique 4.

Parameters

apphandle

Identifies an Isis Application.

actab

Specifies the isisac.tab file name or a string describing the table. If it is a file name, it must be preceded by @. The specification of the extension (*.tab) is not required.

Return Value

Possible return codes are:

- ZERO — Success.
- ERR_FILEMISSING — File missing.
- ERR_FILEREAD — File read error.
- ERR_LLCISISETRAP — Cisis Low Level Error Trap.
- ERR_MEMALLOCAT — Memory Allocation Error.
- ERR_PARAPPHAND — Invalid application handle.
- ERR_PARFILNSIZ — Invalid file name size..
- ERR_PARNULLSTR — String with zero size.

Comments

This function verifies if the presence of *actab.tab* file. Is no table is specified, the standard actab table is used as default.

Example

```
IsisAppAcTab(A, "@c:\\isis\\data\\cdsac.tab");
```

Unchanged since (version)

β 4.0

See Also

IsisAppNew, IsisAppUcTab.

IsisAppDebug

[5] The function's prototypes are described here as they are in the isisdll.h file. The ISIS_DLL prototypes for other languages (Visual Basic, Delphi, and Java) can be seen in Annex 2.

#include<isisdll.h>

long
FAR PASCAL

IsisAppDebug(*long apphandle,*
long debugflag)

The **IsisAppDebug** function specifies the way ISIS_DLL handles run-time error messages. The default value is DEBUG_HARD.

Parameters

apphandle

Identifies an Isis Application.

debugflag

Specifies the error handler method.

Return Value

Possible return codes are:

- ZERO Success.
- ERR_PAROUTRANG Parameter out of range.

Comments

The allowed values for *debugflag* are:

◊ DEBUG_VERY_LIGHT - ISIS_DLL does not display any message nor terminates the execution in case of error.
◊ DEBUG_LIGHT - ISIS_DLL displays an error message and terminates the execution only in case of fatal errors.
◊ DEBUG_HARD - ISIS_DLL displays all error messages but terminates the execution only in case of fatal errors.
◊ DEBUG_VERY_HARD - ISIS_DLL displays an error message and terminates the execution in case of any type of error.

Example

```
long A;

A = IsisAppNew ();
IsisAppDebug (A, DEBUG_HARD);
```

Unchanged since (version)

β 4.0

See Also

IsisAppNew, IsisAppLogFile.

IsisAppDelete

#include<isisdll.h>

long
FAR PASCAL

IsisAppDelete(*long apphandle*)

The **IsisAppDelete** function deletes an Isis Application object and frees all

associated memory.

Parameters

apphandle

Identifies an Isis Application.

Return Value

Possible return codes are:

- ZERO Success.
- ERR_PARAPPHAND Invalid application handle.

Comments

After the deletion, no ISIS_DLL function can be called by the application.

Example

```
long A;

A = IsisAppNew ();
...
IsisAppDelete (A);
```

Unchanged since (version)

β 4.0

See Also

IsisAppNew.

IsisAppLogFile

#include<isisdll.h>

long
FAR PASCAL

IsisAppLogFile(*long apphandle, char *filename*)

The **IsisAppLogFile** function specifies the name of a log file ISIS_DLL uses to log run-time errors.

Parameters

apphandle

Identifies an Isis Application.

filename

Specifies the log file name.

Return Value

Possible return codes are:

- ZERO Success.
- ERR_FILECREATE File creation error.
- ERR_PARFILNSIZ Invalid file name size.

Comments

If no error occurs, the log file is not created. For each run-time error, ISIS_DLL writes a message composed by the ISIS_DLL stack trace and an error code to the log file.

Example

```
long A;

A = IsisAppNew ();
IsisAppLogFile (A, "c:\\app\\applogerr.log");
```

Unchanged since (version) β 7.0

See Also IsisAppNew, IsisAppDebug.

IsisAppNew

#include<isisdll.h>

long
FAR PASCAL

IsisAppNew()

The **IsisAppNew** function creates an ISIS Application. It returns an application handle to create Isis Spaces and displays a message-box window containing the ISIS_DLL copyright notice.

Parameters This function has no parameters.

Return Value Possible values are:

- application handle, if positive
- ERR_MEMALLOCAT Memory Allocation Error.

Comments As an application handle is required to create one or more Isis Spaces, this function implicitly gives the authorization to use the ISIS_DLL. Each application needs only one application handle.

Example

```
long A, H;

A = IsisAppNew ();
H = IsisSpaNew (A);
...
IsisAppDelete (A);
```

Unchanged since (version) β 1.0

See Also IsisSpaNew, IsisAppDelete.

IsisAppParGet

#include<isisdll.h>

long
FAR PASCAL

IsisAppParGet *(long apphandle,*
*char *parinp,*
*char *paroutp,*
long areasize)

The **IsisAppParGet** function retrieves a physical file name from a logical one using the file/path conversion table.

Parameters

apphandle

Identifies an Isis Application.

parinp

Specifies the logical file name.

paroutp

Specifies the area where the physical file name will be copied.

areasize

Specifies the size of the area.

Return Value

Possible values are:

- ZERO Success.
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_PARAPPHAND Invalid application handle.
- ERR_PARNULLPNT NULL pointer.

Comments

The par table is sequence of lines, each line is constituted by:
<logical file name> **=** <physical file name>

This function copies up to *areasize* - 1 bytes into the destination area, truncating the result as necessary.

Example

```
IsisAppParGet (A, "CDS", Areap, 64);
```

Unchanged since (version)

β 5.0

See Also

IsisAppNew, IsisAppParSet

IsisAppParSet

#include<isisdll.h>

long
FAR PASCAL

IsisAppParSet *(long apphandle,*
*char *pararea)*

The **IsisAppParSet** function specifies the file/path conversion table.

Parameters

apphandle

Identifies an Isis Application.

pararea

Specifies the *file.par* file name or a string describing the table. If it is a file name, it must be preceded by @. The specification of the extension is not required.

Return Value

Possible values are:

- ZERO Success.
- ERR_FILEMISSING File missing.
- ERR_FILEWRITE File write error.
- ERR_PARAPPHAND Invalid application handle.

Comments The par table is sequence of lines, each line is constituted by:
<logical file name> **=** <physical file name>

If a NULL string is used as *pararea* parameter, the current par conversion table is no more used by the system.

Example
```
IsisAppParSet (A, "@c:\\bases\\db.par");
```

Unchanged since (version) β 4.0

See Also IsisAppNew, IsisAppParGet

IsisAppUcTab

#include<isisdll.h>

long
FAR PASCAL

IsisAppUcTab*(long apphandle,*
*char *uctab)*

The **IsisAppUcTab** function specifies the Isis uppercase conversion table.

Parameters

apphandle

Identifies an Isis Application.

uctab

Specifies the isisuc.tab file name or a string describing the 256-character table. If it is a file name, it must be preceded by @. The specification of the extension is not required.

Return Value Possible return codes are:

- ZERO Success.
- ERR_FILEMISSING File missing.
- ERR_FILEREAD File read error.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARAPPHAND Invalid application handle.
- ERR_PARFILNSIZ Invalid file name size.
- ERR_PARNULLSTR String with zero size.

Comments This function verifies if the presence of *uctab.tab* file. Is no table is specified, the standard uctab table is used as default.

Example
```
IsisAppUcTab(A, "@c:\\isis\\data\\cdsuc.tab");
```

Unchanged since (version) β 4.0

See Also IsisAppAcTab.

IsisDllVersion

#include<isisdll.h>

float
FAR PASCAL
IsisDllVersion()

The **IsisDllVersion** function retrieves the version identification of the ISIS_DLL.

Parameters This function has no parameters.

Return Value The return value is the version of the ISIS_DLL.

Comments This function may be used to verify compatibility between the DLL version being used and an Isis application.

Example

```
CurrentVersion = IsisDllVersion();

if (CurrentVersion < DesiredVersion)
 printf ("Incorrect ISIS_DLL version");
```

Unchanged since (version) β 1.0

IsisLnkIfLoad

#include<isisdll.h>

long
FAR PASCAL
IsisLnkIfLoad(*long handle*)

The **IsisLnkIfLoad** function loads an Inverted file from the link files (See Inverted File Updating chapter).

Parameters

handle

Identifies the Isis Space.

Return Value Possible values are:

- the number of generated keys.
- ERR_DBDELOCK Data Base access denied (data entry lock).
- ERR_FILEINVERT File does not exist (inverted).
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.

Comments Before using this function, the *ifname.*lk1 and *ifname.*lk2 files must be generated by the IsisRecLnk function.

Example

```
IsisLnkIfLoad (H);
```

Unchanged since (version) β 4.0

See Also IsisRecLnk, IsisLnkSort, IsisLnkIfLoadEx

IsisLnkIfLoadEx

#include<isisdll.h>

long
FAR PASCAL
IsisLnkIfLoadEx(*long handle,*
long reset,
long posts,
long balan)

The **IsisLnkIfLoadEx** function makes a customized inverted file loading from the link files (See Inverted File Updating chapter).

Parameters

handle

Identifies the Isis Space.

reset

(reset = 1), resets the "IF update pending flag"; (reset = 0), the "IF update pending flag" is not removed;

posts

(posts = 1), loads the IF dictionary and IF postings (.ifp), (posts = 0), loads only the IF dictionary with all terms pointing to mfn=1;

balan

(balan = 1), rebalances the IF dictionary, (balan = 0) does not rebalances the IF dictionary.

Return Value Possible values are:

- the number of generated keys.
- ERR_DBDELOCK Data Base access denied (data entry lock).
- ERR_FILEINVERT File does not exist (inverted).
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.

Comments Before using this function, the *ifname.*lk1 and *ifname.*lk2 files must be generated by the IsisRecLnk and IsisLnkSort function.

Example

```
IsisLnkIfLoadEx (H, 1, 0, 0);
```

Unchanged since β 4.0

(version)

See Also IsisRecLnk, IsisLnkSort, IsisLnkIfLoad,

IsisLnkSort

#include<isisdll.h>

long
FAR PASCAL

IsisLnkSort(*long handle*)

The **IsisLnkSort** function generates the *ifname.*lk1 and *ifname.*lk2 files by sorting the link files (*ifname.*ln1 and *ifname.*ln2).

Parameters

handle

Identifies the Isis Space.

Return Value Possible values are:

- ZERO Success.
- ERR_FILEINVERT File does not exist (inverted).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.

Comments The *ifname.*ln1 and *ifname.*ln2 files are sorted by the posting key.

Example

```
IsisRecLnk (H, 1, 99999)
IsisLnkSort (H)
IsisLnkIfLoad (H);
```

Unchanged since (version) β 4.0

See Also IsisRecLnk, IsisLnkIfLoad, IsisLnkIfLoadEx.

IsisRecControlMap

#include<isisdll.h>
#include <isis001.h>

Long
FAR PASCAL

IsisRecControlMap(*long handle,*
*char *ctrl*)

The **IsisRecControlMap** function copies the Control Record (Mfn=0) into an application program area.

Parameters

Handle

Identifies the Isis Space.

Ctrl

Specifies the IsisRecControl structure where the data will be copied.

Return Value Possible values are:

- next master file number
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.

Comments An IsisRecControl structure should be used to avoid memory corruption.

Example

```
IsisRecControl Control;

if (IsisRecControlMap(H, TOCHAR(Control)) > ZERO)
 printf ("Last mfn : %ld" , Control.nxtmfn - 1);
```

Unchanged since (version) β 4.0

See Also IsisSpaMf, IsisRecDirMap, IsisRecLeaderMap, IsisRecRead.

IsisRecCopy

#include<isisdll.h>

Long
FAR PASCAL

IsisRecCopy*(long handle_from,*
long index_from,
long handle_to,
long index_to)

The **IsisRecCopy** function copies a master file record from one shelf to another shelf in the same or a different Isis Space.

Parameters

handle_from

Identifies the source Isis Space.

index_from

Specifies the source record shelf number.

handle_to

Identifies the destination Isis Space.

index_to

Specifies the destination record shelf number.

Return Value Possible values are:

- ZERO Success.
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_FILEMASTER File does not exist (master).
- ERR_PAROUTRANG Parameter out of range.

Comments Source and destination records may but need not be within the same Isis Space.

Example

```
IsisSpaMf (H, "c:\\bases\\cds\\cds");
IsisSpaMf (H1, "c:\\bases\\cds\\cds1");

if (IsisRecRead (H, 0, 1L) == ZERO) {
 IsisRecCopy (H, 0, H1, 0);
 IsisRecWrite(H1, 0);
}
```

Unchanged since (version) β 4.0

See Also IsisRecRead, IsisRecWrite.

IsisRecDirMap

#include<isisdll.h>
#include<isis001.h>

long
FAR PASCAL

IsisRecDirMap(*long handle,*
long index,
long firstpos,
long lastpos,
*char *dir*)

The **IsisRecDirMap** function copies a subset of the directory structure of a master file record into an application program area.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

firstpos

Initial range position in the list of retrieved directory elements (*firstpos* > 0).

lastpos

Last range position in the list of retrieved directory elements (lastpos >= firstpos).

dir

Specifies an array of IsisRecDir structures where the data will be copied.

Return Value Possible values are:

- number of bytes copied
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Isis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.

- ERR_PAROUTRANG Parameter out of range.

Comments An array of IsisRecDir structures with an appropriate size should be used in order to avoid memory corruption.

Example

```
void showdir (long handle, long mfn)
{
 IsisRecDir  *Dirp;
 long         Cont;

 IsisRecRead(handle, 0, mfn);

 IsisRecLeader Leader;
 IsisRecLeaderMap (handle, 0, TOCHAR(Leader));

 // Allocate the array of IsisRecDir structure
 Dirp = new IsisRecDir [Leader.nvf];
 if (!Dirp) exit(EXIT_FAILURE);

// Copy directory.
 IsisRecDirMap(handle, 0, 1, Leader.nvf , (char*)Dirp);
// Obs: Using Visual Basic and/or Delphi one should use
// the first array element address parameter
// IsisRecDirMap(handle, 0, 1, Leader.nvf, Dirp[0]);

 // Show directory.
 for(Cont=0; Cont< Leader.nvf; Cont++)
 {
  printf ("tag = %ld \n", Dirp[Count].tag);
  printf ("pos = %ld \n", Dirp[Count].pos);
  printf ("len = %ld \n", Dirp[Count].len);
 }

 delete[] Dirp;
}
```

Unchanged since (version) β 5.0

See Also IsisRecRead, IsisRecControlMap, IsisRecLeaderMap.

IsisRecDummy

#include<isisdll.h>

Long
FAR PASCAL

IsisRecDummy (*long handle,*
long index)

The **IsisRecDummy** function creates an empty record in a given shelf.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value Possible values are:

- ZERO Success.
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments The dummy record is generally used as a temporary record.
Its master file number is -1L. When saved into a master file, it will be assigned the next available master file number.
The IsisRecDummy function is only necessary after a record shelf has been used because when an Isis space is created (IsisSpaNew) an empty record is also created in the shelf number 0.

Example

```
IsisRecRead (H, 0, 32);

// Create a new record in a shelf that already has a
record.
IsisRecDummy (H, 0);
IsisRecFieldUpdate (H, 0, "a10#new field#");
```

Unchanged since (version) β 4.0

See Also IsisSpaMf, IsisRecNew, IsisRecNewLock, IsisRecWrite.

IsisRecDump

#include<isisdll.h>

Long
FAR PASCAL

IsisRecDump(*long handle,*
long index,
*char *dump,*
long areasize)

The **IsisRecDump** function dumps into an application program area, all the fields of the record stored in a given shelf. Each field contents are delimited by an initial and a final field mark. The initial field mark is composed by the field tag preceded by the start tag delimiter (<) and followed by the end tag delimiter (>). The end field mark is composed by the start tag delimiter (<), the slash character (/) and the tag followed by the end tag delimiter (>):

<tag1>field1</tag1>
<tag2>field2</tag2>
<tag3>field3</tag3>
...

Parameters *handle*

Identifies the Isis Space.

index

Specifies the record shelf number.

dump

Specifies the area where the data will be dumped.

areasize

Specifies the size of the area.

Return Value Possible values are:

- number of bytes copied
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments This function copies up to *areasize* - 1 bytes into the destination area, truncating the result as necessary.
The start and end tag delimiters can be changed using the IsisSpaRecDelim function.

Example

```
if (IsisRecRead (H, 0, 15) == ZER0)
{
 IsisRecDump (H, 0, Area, 2000);
 printf ("%s \n", Area);
}
```

Unchanged since (version) β 5.0

See Also IsisRecRead, IsisRecFormat, IsisRecFormatEx, IsisSpaRecDelim.

IsisRecField

#include<isisdll.h>

long
FAR PASCAL

IsisRecField(*long handle,*
long index,
long tag,
long occ,
*char *field,*
long areasize)

The **IsisRecField** function copies the contents of a field occurrence to an application program area.

Parameters *handle*

Identifies the Isis Space.

index

Specifies the record shelf number.

tag

Specifies the field tag number (tag > 0).

occ

Specifies the field occurrence number (*occ* > 0).

field

Specifies the pre-allocated area where the data will be copied.

areasize

Specifies the size of the area.

Return Value

Possible values are:

- number of bytes copied
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments

If field does not exist, zero is returned.

This function copies up to *areasize* - 1 bytes into the destination area, truncating the result as necessary.

Example

```
if (IsisRecRead (H, 0, 12) == ZERO)
{
 IsisRecField(H, 0, 12, 1, area12, area12size);
 printf ("field : %s \n", area12);
}
```

Unchanged since (version)

β 5.0

See Also

IsisRecRead, IsisRecFieldN, IsisRecFieldOcc, IsisRecSubField, IsisRecSubFieldEx.

IsisRecFieldN

#include<isisdll.h>

long
FAR PASCAL

IsisRecFieldN(*long handle,*
long index,
long nth,
*char *field,*
long areasize)

The **IsisRecFieldN** function copies the contents of the *nth*-entry directory field to an application program area.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

nth

Specifies the record directory entry (directory entries are numbered starting from zero).

field

Specifies the pre-allocated area where the data will be copied.

areasize

Specifies the size of the area.

Return Value

Possible values are:

- number of bytes copied
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments

This function copies up to *areasize* - 1 bytes into the destination area, truncating the result as necessary.

Example

```
if (IsisRecRead (H, 0, 12) == ZERO)
 if (IsisRecFieldN(H, 0, 1, area12, area12size) > ZERO)
  printf ("field : %s \n", area12);
```

Unchanged since (version)

β 5.0

See Also

IsisRecRead, IsisRecField, IsisRecFieldOcc, IsisRecSubField, IsisRecSubFieldEx.

IsisRecFieldOcc

#include<isisdll.h>

Long
FAR PASCAL

IsisRecFieldOcc(*long handle,*
long index,
long tag)

The **IsisRecFieldOcc** returns the number of occurrences of a given field.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

tag

Specifies the tag number of the desired field (tag > 0).

Return Value

Possible values are:

- number of occurrences of the field

- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments If the desired field is not present, the return value will be zero.

Example

```
if (IsisRecRead (H, 0, 12) == ZERO)
 printf ("occ num : %ld \n", IsisRecFieldOcc(H, 0, 1));
```

Unchanged since (version) β 4.0

See Also IsisRecRead, IsisRecField, IsisRecFieldN, IsisRecSubField, IsisRecSubFieldEx.

IsisRecFieldUpdate

#include<isisdll.h>

Long
FAR PASCAL

IsisRecFieldUpdate(*long handle,*
long index,
*char *fldupd*)

The **IsisRecFieldUpdate** function executes one or more commands to add and/or delete fields from a master file record.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

fldupd

Specifies the commands to be executed by this function.

Return Value Possible values are:

- ZERO Success.
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARFLDUPSYNT Syntax Error (field update).
- ERR_PAROUTRANG Parameter out of range.

Comments The *fldupd* string should contains one or more of the following commands before calling **IsisRecFieldUpdate**:

d.	Makes the record logically deleted.
d*	Deletes all present fields.
d*tt*	Deletes all occurrences of field *tt.*
d*tt*/occ	Deletes occurrence *occ* of field *tt.*
a*tt#str#*	Adds string *str* as a new occurrence of field *tt.*

h*tt n str*	Adds string *str*, *n* bytes long, as a new occurrence of field *tt.*
=n	Assigns *n* to MFN
s	Sorts the directory entries by tag value

To keep the updated fields it is necessary to save the record calling the IsisRecWrite function.

Example

```
IsisRecControlMap(H, Control);

for (Mfn = 1; Mfn <= Control.nxtmfn - 1; Mfn++)
 if (IsisRecRead(H, 0, Mfn) == ZERO)
 {
  // deletes all record fields and then add the field
//  number one.
  IsisRecFieldUpdate(H, 0, "d* a1!CDS!");
  IsisRecWrite(H, 0);
 }
```

Unchanged since (version) β 4.0

See Also IsisRecRead, IsisRecWrite, IsisRecField.

IsisRecFormat

#include<isisdll.h>

long
FAR PASCAL

IsisRecFormat(*long handle,*
long index,
*char *area,*
long areasize)

The **IsisRecFormat** function formats a record into an application program area, according to a CDS/ISIS format specification. The function returns the number of characters formatted. The format is defined by a call to the IsisSpaPft function.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

area

Specifies the pre-allocated program area for output of the format.

areasize

Specifies the size of the area.

Return Value Possible values are:

- number of bytes copied
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.

- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARFMTSYNT Syntax Error (format).
- ERR_PAROUTRANG Parameter out of range.

Comments This function copies up to *areasize* - 1 bytes into the destination area, truncating the result as necessary. If the format specification used as argument in the IsisSpaPft function is an empty string, the output of the format will be a record dump.

Example

```
IsisSpaPft(H, "@c:\\bases\\cds\\cds");

Areaf = new char[AreafSize];
R = ZERO;
Mfn = 1;

while (R <> ERR_RECEOF) {
 if ((R = IsisRecRead(H, 0, Mfn++)) == ZERO)
 {
  IsisRecFormat(H, 0, Areaf, AreafSize);
  printf ("%s", Areaf);
 }
}

delete[] Areaf;
```

Unchanged since (version) β 7.0

See Also IsisSpaPft, IsisSpaMf, IsisSpaIf, IsisRecRead, IsisRecFormatEx, IsisRecDump.

IsisRecFormatEx

#include<isisdll.h>

long
FAR PASCAL

IsisRecFormatEx(*long handle,*
long index,
long linesize,
*char *area,*
long areasize)

The **IsisRecFormatEx** function formats a record into an application program area, according to a CDS/ISIS format specification. The formatted area is broken in lines with a specified maximum length. The function returns the number of characters formatted.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

linesize

Specifies the maximum line length (linesize > 0).

area

Specifies the pre-allocated program area for output of the format.

areasize

Specifies the size of the area.

Return Value

Possible values are:

- number of bytes copied
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARFMTSYNT Syntax Error (format).
- ERR_PAROUTRANG Parameter out of range.

Comments

This function copies up to *areasize* - 1 bytes into the destination area, truncating the result as necessary.

Example

```
IsisSpaPft(H, "@c:\\bases\\cds\\cds");

Areaf = new char[AreafSize];
R = ZERO;
Mfn = 1;

while (R <> ERR_RECEOF) {
 if ((R = IsisRecRead(H, 0, Mfn++)) == ZERO)
 {
  IsisRecFormatEx(H, 0, 60, Areaf, AreafSize);
  printf ("%s", Areaf);
 }
}

delete[] Areaf;
```

Unchanged since (version)

β 7.0

See Also

IsisSpaPft, IsisSpaMf, IsisSpaIf, IsisRecRead, IsisRecFormat, IsisRecFormatEx, IsisRecDump.

IsisRecIfUpdate

#include<isisdll.h>

Long
FAR PASCAL

IsisRecIfUpdate(long *handle,*
long mfn)

The **IsisRecIfUpdate** function performs the inversion of record *mfn* in the master file specified by the Isis Space *handle*. The field select table used in the inversion process is defined by IsisSpaFst function.

Parameters

handle

Identifies the Isis Space.

mfn

Specifies the master file record number (mfn > 0).

Return Value

Possible values are:

- ZERO — Success.
- ERR_DBDELOCK — Data Base access denied (data entry lock).
- ERR_FILEFST — File does not exist (fst).
- ERR_FILEINVERT — File does not exist (inverted).
- ERR_FILEMASTER — File does not exist (master).
- ERR_LLCISISETRAP — Cisis Low Level Error Trap.
- ERR_MEMALLOCAT — Memory Allocation Error.
- ERR_PAROUTRANG — Parameter out of range.

Comments

The **IsisRecIfUpdate** function reads the record from the master file, updates the inverted file and resets the record inverted file update pending flag. Note however that IsisRecIfUpdate does not require this flag to be turned on in order to invert the record.

Example

```
Mfn = 10;

IsisRecIfUpdate(H, Mfn);
```

Unchanged since (version)

β 4.0

See Also

IsisSpaMf, IsisSpaIf, IsisSpaFst, IsisRecIfUpdateEx.

IsisRecIfUpdateEx

#include<isisdll.h>

Long
FAR PASCAL

IsisRecIfUpdateEx(long *handle,*
long beginmfn,
long endmfn,
long keeppending)

The **IsisRecIfUpdateEx** function performs the inversion of a range of record mfns in the master file specified by the Isis Space *handle*. The field select table used in the inversion process is defined by IsisSpaFst function.

Parameters

handle

Identifies the Isis Space.

beginmfn

Specifies the initial master file record number (begin mfn > 0).

endmfn

Specifies the end master file record number (endmfn >= beginmfn).

keeppending

Specifies if the "inverted file update pending flag" will be kept (0 - does not keep the flag set, 1 - keeps the flag set).

Return Value Possible values are:

- ZERO Success.
- ERR_DBDELOCK Data Base access denied (data entry lock).
- ERR_FILEFST File does not exist (fst).
- ERR_FILEINVERT File does not exist (inverted).
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments The **IsisRecIfUpdateEx** function reads each record of the range from the master file, updates the inverted file and resets the record inverted file update pending flag. Note however that IsisRecIfUpdate does not require this flag to be turned on in order to invert the record.

Example

```
MfnB = 10;
MfnE = 20

IsisRecIfUpdateEx(H, MfnB, MfnE, 0);
```

Unchanged since (version) β 4.0

See Also IsisSpaMf, IsisSpaIf, IsisSpaFst, IsisRecIfUpdate.

IsisRecIsoRead

#include<isisdll.h>

long
FAR PASCAL

IsisRecIsoRead(*long handle,*
long index)

The **IsisRecIsoRead** function loads a record from an ISO 2709 format file into a master file record shelf.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value Possible values are:

- record size
- ERR_FILEISO File does not exist (ISO).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments An ISO 2709 file must be specified with IsisSpaIsoIn function before calling IsisRecIsoRead.

Example

```
IsisSpaIsoIn(H, "c:\\isis\\data\\cds");
while ((RC=IsisRecIsoRead(Spahandle, 0)) > 0)
{
  IsisRecWrite(Spahandle, 0);
  printf ("rc : %ld importing mfn : %ld \n", RC, Mfn++);
}
```

Unchanged since (version) β 4.0

See Also IsisSpaIsoIn, IsisSpaIsoOut, IsisSpaIsoDelim, IsisRecIsoWrite.

IsisRecIsoWrite

#include<isisdll.h>

long
FAR PASCAL

IsisRecIsoWrite(*long handle,*
long index)

The **IsisRecIsoWrite** function writes a record from a shelf index into an ISO 2709 output file.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value Possible values are:

- record size
- ERR_FILEISO File does not exist (ISO).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments An ISO 2709 export file must be specified with IsisSpaIsoOut function before calling IsisRecIsoWrite. The default field and record character delimiter is the character #. To change the field and/or record character delimiter use IsisSpaIsoDelim function.

Example

```
while ((RC=IsisRecRead(Spahandle, 0, Mfn++)) !=
ERR_RECEOF)
{
  printf ("rc : %ld exporting mfn : %ld \n", RC, Mfn);
  if (RC == ZERO)
    IsisRecIsoWrite(Spahandle, 0);
}
```

Unchanged since (version) β 4.0

See Also IsisSpaIsoIn, IsisRecIsoRead, IsisSpaIsoOut, IsisSpaIsoDelim.

IsisRecLeaderMap

#include<isisdll.h>
#include<isis001.h>

long
FAR PASCAL

IsisRecLeaderMap(*long handle,*
long index,
*char *leader)*

The **IsisRecLeaderMap** function copies the leader of the record at shelf *index* into an application program area.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

leader

Specifies the pre-allocated area where the data will be copied.

Return Value

Possible values are:

- number of variable fields
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments

An IsisRecLeader structure should be used to avoid memory corruption.

Example

```
IsisRecControl Control;
IsisRecLeader Leader;

Min = 99999;
Max = 0;
Avg = 0;

IsisRecControlMap(H, Control);

for (Mfn = 1; Mfn <= Control.nxtmfn - 1; Mfn++)
 if (IsisRecRead(H, 0, Mfn) == ZERO)
 {
  IsisRecLeaderMap(H, 0, Leader);
  if (Leader.mfrl < Min) Min = Leader.mfrl;
  if (Leader.mfrl > Max) Max = Leader.mfrl;
  Avg = Avg + Leader.mfrl;
 }

Avg = Avg / (Control.nxtmfn - 1);

printf ("Min:%ld Max:%ld Avg:%f", Min, Max, Avg);
```

Unchanged since (version) β 4.0

See Also IsisRecRead, IsisRecDirMap, IsisRecControlMap, IsisRecDump.

IsisRecLnk

#include<isisdll.h>

long
FAR PASCAL

IsisRecLnk*(long handle,*
long beginmfn,
long endmfn)

The **IsisRecLnk** function generates the inverted link files *.ln1 and *.ln2 from a range of master record.

Parameters

handle

Identifies the Isis Space.

beginmfn

Specifies the initial master file record number (beginmfn > 0).

endmfn

Specifies the end master file record number (endmfn >= beginmfn).

Return Value Possible values are:

- The number of generated links.
- ERR_FILEFST File does not exist (fst).
- ERR_FILEINVERT File does not exist (inverted).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments The generated links must then be sorted (IsisLnkSort) and loaded (IsisLnkIfLoad) before they can be retrieved.

Example

```
IsisRecLnk(H, 1, 10);
```

Unchanged since (version) β 4.0

See Also IsisLnkSort, IsisLnkIfLoad

IsisRecLockRecall

#include<isisdll.h>

Long
FAR PASCAL

IsisRecLockRecall*(long handle,*
long index,
long mfn,
long tag,
char *password)

The **IsisRecLockRecall** function reads a master file record, with master file number - mfn, and retrieves the control of its lock via a password that should be stored in a field of the record.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

mfn

Specifies the master file record number.

tag

Specifies the tag of the record that has the password.

password

Specifies the password to be stored into the record field.

Return Value

Possible values are:

- ZERO — Success.
- ERR_DBEWLOCK — Data Base access denied (probably exclusive write lock)
- ERR_FILEMASTER — File does not exist (master).
- ERR_LLCISISETRAP — Cisis Low Level Error Trap.
- ERR_MEMALLOCAT — Memory Allocation Error.
- ERR_RECEOF — Record eof: found eof in data base.
- ERR_RECLOCKED — Record locked.
- ERR_RECLOGIDEL — Record logically deleted.
- ERR_RECPHYSDEL — Record physically deleted.

Comments

This function is supposed to be used by programs that know in advance, they will finish without releasing the lock of a record.

Example

```
IsisRecLockRecall(H, 0, 15, 24, “psd_15”);
```

Unchanged since (version)

β 6.0

See Also

IsisSpaMf, IsisRecReadLock, IsisRecNewLock, IsisRecUnlockForce.

IsisRecMerge

#include<isisdll.h>

long

FAR PASCAL

IsisRecMerge(*long handle_from,*
long index_from,
long handle_to,
long index_to)

The **IsisRecMerge** function adds the fields of a master file record to another master file record stored in the same or a different Isis Space.

Parameters

handle_from

Identifies the source Isis Space.

index_from

Specifies the source record shelf number.

handle_to

Identifies the destination Isis Space.

index_to

Specifies the destination record shelf number.

Return Value

Possible values are:

- ZERO Success.
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments

Source and destination records may but need not be within the same Isis Space. It is possible to append one record to itself.

Example

```
IsisSpaMf(H, "c:\\isis\\data\\cds")

if (IsisRecRead(H, 0, Mfn) == ZERO)
 IsisRecMerge(H, 0, H, 0);
```

Unchanged since (version)

β 4.0

See Also

IsisRecRead, IsisRecWrite, IsisRecCopy.

IsisRecMfn

#include<isisdll.h>

long
FAR PASCAL

IsisRecMfn(long *handle,*
long index)

The **IsisRecMfn** function retrieves the master file number of the record at shelf *index*.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value Possible values are:

- master file record number
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments If the record is a dummy record, its mfn is -1L.

Example

```
printf ("Shelf %ld has record %ld", S, IsisRecMfn(H, S));
```

Unchanged since (version) β 4.0

See Also IsisRecRead, IsisRecMfnChange.

IsisRecMfnChange

#include<isisdll.h>

long
FAR PASCAL

IsisRecMfnChange(*long handle,*
long index,
long nmfn)

The **IsisRecMfnChange** function assigns a new master file number (Mfn) to the record at shelf *index*.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

nmfn

Specifies the new master file number (mfn > 0).

Return Value Possible values are:

- ZERO Success.
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments This function should be used with caution in order to avoid misassignment of MFNs.This function only changes the mfn in memory, to change it in the data base it is necessary to save the record using the IsisRecWrite function.

Example

```
Mfn = 10;
Rc = ZERO;

while (Rc != ERR_RECEOF)
 if ((Rc = IsisRecRead (H, 0, Mfn)) == ZERO)
 {
  IsisRecMfnChange(H, 0, Mfn-5);
  IsisRecWrite (H, 0);
  Mfn ++;
 }
```

Unchanged since (version) β 4.0

See Also IsisRecRead, IsisRecWrite, IsisRecMfn.

IsisRecNew

#include<isisdll.h>

long
FAR PASCAL

IsisRecNew(long *handle,*
long *index*)

The **IsisRecNew** function creates a record in shelf *index* and in the master file with a new Mfn. The new record's Mfn is one higher than the largest Mfn currently stored in the data base.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value Possible values are:

- master file record number
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments To create a temporary record, i.e., a record that will not actually be written to the master file, use the IsisRecDummy function. To save the new record in the data base use the IsisRecWrite function.

Example

```
if (IsisRecNew (H, 0) > ZERO)
{
 IsisRecFieldUpdate (H, 0, "a10$new record$");
 IsisRecWrite (H, 0);
}
```

Unchanged since β 4.0

(version)

See Also IsisSpaMf, IsisRecDummy, IsisRecNewLock, IsisRecWrite.

IsisRecNewLock

#include<isisdll.h>

long
FAR PASCAL

IsisRecNewLock(long *handle,*
long *index*)

The **IsisRecNewLock** function creates a record at shelf *index* and in the master file, with a new Mfn, and keeps it locked.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value Possible values are:

- master file record number
- ERR_DBEWLOCK Data Base access denied (probably exclusive write lock).
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments This function is used in concurrent operation mode.

Example

```
if (IsisRecNewLock (H, 0) > ZERO)
{
 IsisRecFieldUpdate (H, 0, "a10#new record#");
 IsisRecWriteUnlock (H, 0);
}
```

Unchanged since (version) β 4.0

See Also IsisSpaMf, IsisRecNew, IsisRecDummy, IsisRecWrite, IsisRecWriteUnlock.

IsisRecNvf

#include<isisdll.h>

long
FAR PASCAL

IsisRecNvf(long *handle,*

long index)

The **IsisRecNvf** function returns the number of variable fields in the record at shelf *index*.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value

Possible values are:

- number of variable fields
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments

If the record has no fields, this function will return zero.

Example

```
printf("Nvf: %ld", IsisRecNvf (H, 0));
```

Unchanged since (version)

β 4.0

See Also

IsisRecRead, IsisRecLeaderMap.

IsisRecRead

#include<isisdll.h>

long
FAR PASCAL

IsisRecRead(*long handle,*
long index,
long mfn)

The **IsisRecRead** function reads a master file record with master file number *mfn* and copies it into the shelf *index.*

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

mfn

Specifies the master file number of the record (mfn > 0).

Return Value

Possible values are:

- ZERO Success.
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.

- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.
- ERR_RECEOF Record eof: found eof in data base.
- ERR_RECLOCKED Record locked.
- ERR_RECLOGIDEL Record logically deleted.
- ERR_RECPHYSDEL Record physically deleted.

Comments A call to this function deletes the previous contents of the shelf *index*. This function should only be used to read data records. To access Control Record data use IsisRecControl.

Example

```
if (IsisRecRead(H, 0, Mfn) == ZERO)
{
 IsisRecDump (H, 0, Area, 3000);
 printf (Area);
}
```

Unchanged since (version) β 4.0

See Also IsisSpaMf, IsisRecReadLock, IsisRecControl, IsisRecFieldUpdate.

IsisRecReadLock

#include<isisdll.h>

long
FAR PASCAL

IsisRecReadLock(*long handle,*
long index,
long mfn)

The **IsisRecReadLock** function reads a master file record with master file record number *mfn* into the shelf *index* and keeps it locked in the data base.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

mfn

Specifies the master file number of the record (mfn > 0).

Return Value Possible values are:

- ZERO Success.
- ERR_DBEWLOCK Data Base access denied (probably exclusive write lock).
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.
- ERR_RECEOF Record eof: found eof in data base.
- ERR_RECLOCKED Record locked.

- ERR_RECLOGIDEL Record logically deleted.
- ERR_RECPHYSDEL Record physically deleted.

Comments A call to this function deletes the previous contents of the shelf *index*.

Example

```
if (IsisRecReadLock (H, 0, 100) > ZERO)
{
 IsisRecFieldUpdate (H, 0, "a10#new record#");
 IsisRecWriteUnlock (H, 0);
}
```

Unchanged since (version) β 4.0

See Also IsisSpaMf, IsisRecRead, IsisRecControl, IsisRecFieldUpdate.

IsisRecShelfSize

#include<isisdll.h>

long
FAR PASCAL

IsisRecShelfSize*(long handle,*
long index,
long mem)

The **IsisRecShelfSize** function changes the size of a master file record shelf.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

mem

Specifies the number of bytes to be allocated (mem > 8192 bytes).

Return Value Possible values are:

- ZERO Success.
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments This function should be used when the record length is different from 30000 bytes (MAXMFRL), the default record length.

Example

```
r = IsisRecShelfSize(H, I, 48000);
```

Unchanged since (version) β 4.0

See Also IsisSpaNew, IsisSpaRecShelves, IsisTrmShelfSize.

IsisRecSubField

#include<isisdll.h>

long
FAR PASCAL

IsisRecSubField(*long handle,*
long index,
long tag,
long occ,
*char *subfield,*
*char *subfieldarea,*
long areasize)

The **IsisRecSubField** function copies the contents of a specified subfield to an application program area.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

tag

Specifies the field tag number (tag > 0).

occ

Specifies the field occurrence number (*occ* > 0).

subfield

Specifies the subfield.

subfieldarea

Specifies the area where the contents of the requested subfield will be copied.

areasize

Specifies the size of the area.

Return Value

Possible values are:

- number of bytes copied
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments

If subfield does not exist, zero is returned.

This function copies up to *areasize* - 1 bytes into the destination area, truncating the result as necessary.

Example

```
r = IsisRecSubField(H, 0, 12, 1, "a", area12, area12size);
```

Unchanged since β 5.0

(version)

See Also IsisRecRead, IsisRecField, IsisRecSubFieldEx.

IsisRecSubFieldEx

#include<isisdll.h>

long
FAR PASCAL

IsisRecSubFieldEx(*long handle,*
long index,
long tag,
long fldocc,
*char *subfield,*
long subfldocc
*char *subfieldarea,*
long areasize)

The **IsisRecSubFieldEx** function copies the contents of an occurrence of a specified subfield to an application program area.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

tag

Specifies the field tag number (tag > 0).

fldocc

Specifies the field occurrence number (*fldocc* > 0).

subfield

Specifies the subfield.

subfldocc

Specifies the subfield occurrence number (*subfldocc* > 0).

subfieldarea

Specifies the area where the contents of the requested subfield will be copied.

areasize

Specifies the size of the area.

Return Value Possible values are:

- number of bytes copied
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments If subfield does not exist, zero is returned.

This function copies up to *areasize* - 1 bytes into the destination area, truncating the result as necessary.

Example

```
r = IsisRecFieldEx(H, 0, 12, 1, "a", 2, area12);
```

Unchanged since (version) β 5.0

See Also IsisRecRead, IsisRecField, IsisRecSubField.

IsisRecUndelete

#include<isisdll.h>

long
FAR PASCAL
IsisRecUndelete(*long handle,*
long index)

The **IsisRecUndelete** function undeletes a logically deleted record in the shelf *index*.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value Possible values are:

- ZERO — Success.
- ERR_FILEMASTER — File does not exist (master).
- ERR_LLCISISETRAP — Cisis Low Level Error Trap.
- ERR_MEMALLOCAT — Memory Allocation Error.
- ERR_PAROUTRANG — Parameter out of range.

Comments The undeleted record may be saved into the data base through IsisRecWrite or IsisRecWriteLock as appropriate.

Example

```
if (IsisRecRead (H, 0, Mfn) == ERR_RECLOGIDEL)
{
 IsisRecUndelete (H, 0);
 IsisRecWrite (H, 0);
}
```

Unchanged since (version) β 4.0

See Also IsisRecRead, IsisRecFieldUpdate, IsisRecWrite.

IsisRecUnlock

#include<isisdll.h>

long
FAR PASCAL

IsisRecUnlock(*long handle,*
long index)

The **IsisRecUnlock** function unlocks the master file record at the shelf *index,* provided that the calling application owns the lock.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value

Possible values are:

- ZERO — Success.
- ERR_DBEWLOCK — Data Base access denied (probably exclusive write lock).
- ERR_FILEMASTER — File does not exist (master).
- ERR_LLCISISETRAP — Cisis Low Level Error Trap.
- ERR_MEMALLOCAT — Memory Allocation Error.
- ERR_PAROUTRANG — Parameter out of range.
- ERR_RECEOF — Record eof: found eof in data base.
- ERR_RECLOCKED — Record locked.
- ERR_RECLOGIDEL — Record logically deleted.
- ERR_RECPHYSDEL — Record physically deleted.

Comments

A call to this function does not change the contents of the shelf *index*, nor writes the record into the master file. It simply removes the record lock. To unlock the record in the disk file it is necessary to call IsisRecWriteUnlock.

Example

```
if (IsisRecRead (H, I, 13) == ERR_RECLOCKED)
 IsisRecUnlock (H, I);
```

Unchanged since (version)

β 4.0

See Also

IsisRecNewLock, IsisRecReadLock, IsisRecWriteUnlock, IsisRecUnlockForce.

IsisRecUnlockForce

#include<isisdll.h>

long
FAR PASCAL

IsisRecUnlockForce(*long handle,*
long index)

The **IsisRecUnlockForce** function unlocks the master file record at the shelf

index, even if the calling application does not own the lock.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value

Possible values are:

- ZERO — Success.
- ERR_DBEWLOCK — Data Base access denied (probably exclusive write lock).
- ERR_FILEMASTER — File does not exist (master).
- ERR_LLCISISETRAP — Cisis Low Level Error Trap.
- ERR_MEMALLOCAT — Memory Allocation Error.
- ERR_PAROUTRANG — Parameter out of range.
- ERR_RECEOF — Record eof: found eof in data base.
- ERR_RECLOGIDEL — Record logically deleted.
- ERR_RECPHYSDEL — Record physically deleted.

Comments

A call to this function does not change the contents of the shelf *index*, nor writes the record into the master file. It simply removes the record lock. To unlock the record in the disk file it is necessary to call IsisRecWriteUnlock.

Example

```
IsisRecUnlockForce (H, I);
```

Unchanged since (version)

β 4.0

See Also

IsisRecNewLock, IsisRecReadLock, IsisRecWriteUnlock, IsisRecUnlock.

IsisRecUpdate

#include<isisdll.h>

long
FAR PASCAL

IsisRecUpdate(long *handle,*
long index,
*char *area*)

The **IsisRecUpdate** function processes a record update string and updates the contents of the record at shelf *index*. Field contents are delimited by an initial and a final field marks. The initial field mark is composed by the field tag preceded by the start tag delimiter (<) and followed by the end tag delimiter (>). The end field mark is composed by the start tag delimiter (<), the slash character (/), and the field tag followed by the end tag delimiter (>):

<tag1>field1</tag1>
<tag2>field2</tag2>
<tag3>field3</tag3>
...

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

area

Specifies the area from where data will be moved.

Return Value

Possible values are:

- ZERO — Success.
- ERR_FILEMASTER — File does not exist (master).
- ERR_LLCISISETRAP — Cisis Low Level Error Trap.
- ERR_MEMALLOCAT — Memory Allocation Error.
- ERR_PARNULLSTR — String with zero size.
- ERR_PAROUTRANG — Parameter out of range.
- ERR_PARUPDSYNT — Syntax Error (record update).

Comments

This function recognizes field tags and contents marked using the delimiter character specified via the function IsisRecDelim. To save the contents of the record in the data base it is necessary to call the IsisRecWrite function.

Example

```
Mfn = 3
r = IsisRecRead(H, 0, Mfn);
r = IsisRecDump(H, 0, Area, 2000);
r = IsisRecUpdate(H, 0, Area);
```

Unchanged since (version)

β 4.0

See Also

IsisRecRead, IsisRecDump, IsisRecFieldUpdate, IsisRecWrite.

IsisRecWrite

#include<isisdll.h>

long
FAR PASCAL

IsisRecWrite(*long handle,*
long index)

The **IsisRecWrite** function writes a master file record from shelf *index* into the master file.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value

Possible values are:

- ZERO — Success.
- ERR_DBEWLOCK — Data Base access denied (probably

exclusive write lock).
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.
- ERR_RECNOTNORM Record condition is not RCNORMAL.

Comments If you write a dummy record (Mfn = -1L), the master file number (Mfn) is changed to the next Mfn available.

Example
```
IsisRecUpdate (H, 0,"d*");
IsisRecWrite(H, 0);
```

Unchanged since (version) β 4.0

See Also IsisRecDummy, IsisRecNew, IsisRecRead, IsisRecFieldUpdate, IsisRecUpdate, IsisRecWriteLock, IsisRecWriteUnlock.

IsisRecWriteLock

#include<isisdll.h>

long
FAR PASCAL
IsisRecWriteLock(*long handle,*
long index)

The **IsisRecWriteLock** function writes the master file record from shelf *index* into a master file and keeps it locked.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value Possible values are:

- ZERO Success.
- ERR_DBEWLOCK Data Base access denied (probably exclusive write lock).
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments If you write a dummy record (Mfn = -1L), the master file number (Mfn) is changed to the next Mfn available.

Example
```
IsisRecDummy (H, 0);
IsisRecWriteLock (H, 0);
```

Unchanged since (version) β 4.0

See Also IsisRecNewLock, IsisRecReadLock, IsisRecFieldUpdate, IsisRecUpdate,IsisRecWrite, IsisRecWriteUnlock.

IsisRecWriteUnlock

#include<isisdll.h>

long
FAR PASCAL

IsisRecWriteUnlock(*long handle,*
long index)

The **IsisRecWriteUnlock** function writes the master file record from shelf *index* into a master file and unlocks it.

Parameters

handle

Identifies the Isis Space.

index

Specifies the record shelf number.

Return Value Possible values are:

- ZERO Success.
- ERR_DBEWLOCK Data Base access denied (probably exclusive write lock).
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments If you write a dummy record (Mfn = -1L), the master file number (Mfn) is changed to the next Mfn available.

Example

```
IsisRecNewLock (H, I);
IsisRecUpdate (H, I, "a3#new record#");
IsisRecWriteUnlock (H, I);
```

Unchanged since (version) β 4.0

See Also IsisRecNewLock, IsisRecReadLock, IsisRecFieldUpdate, IsisRecUpdate,, IsisRecWrite, IsisRecWriteLock.

IsisSpaDb

#include<isisdll.h>

long
FAR PASCAL

IsisSpaDb(*long handle,*

*char *dbname*)

The **IsisSpaDb** function defines all the files of a data base (*.mst, *.xrf, *.cnt, *.n01, *.n02, *.l01, *.l02, *.ifp, *.pft, *.fst and *.stw) provided they have the same name as the data base. This function is useful when you want to define all the files with a single function call. If you do not need all the data base files simultaneously, you may want to use individual calls to the functions IsisSpaMf, IsisSpaIf, IsisSpaPft, IsisSpaFst, IsisSpaStw, as required.

Parameters

handle

Identifies the Isis Space.

dbname

Specifies the data base name. Extension is not required.

Return Value

- ZERO Success.
- ERR_FILEFST File does not exist (fst).
- ERR_FILEINVERT File does not exist (inverted).
- ERR_FILEPFT File does not exist (pft).
- ERR_FILEREAD File read error.
- ERR_FILESTW File does not exist (stw).
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARFILNSIZ Invalid file name size.
- ERR_PARFMTSYNT Syntax Error (format).

Comments

If some of these files do not exist or have a different name, this function will return an error code. Use IsisSpaHeaderMap to verify which files are missing.

Example

```
A = IsisAppNew();
H = IsisSpaNew(A);
R = IsisSpaDb (H, "c:\\isis\\data\\cds");
...
R = IsisAppDelete(A);
```

Unchanged since (version)

β 4.0

See Also

IsisSpaNew, IsisSpaMf, IsisSpaIf, IsisSpaPft, IsisSpaFst, IsisSpaStw.

IsisSpaDelete

#include<isisdll.h>

Long
FAR PASCAL
IsisSpaDelete(*long handle*)

The **IsisSpaDelete** function deletes an Isis Space created with the function IsisSpaNew and frees all associated memory.

Parameters

Handle

Identifies the Isis Space.

Return Value The return code is always ZERO.

Comments The order of deletion of created Isis Spaces is not important. Note that IsisAppDelete deletes all Isis Spaces not previously deleted by IsisSpaDelete. After being deleted, an Isis Space can no longer be used.

Example

```
IsisSpaNew (H);
IsisSpaNew (H1);
...
IsisSpaDelete(H);
IsisSpaDelete(H1);
IsisAppDelete(A);
```

Unchanged since (version) β 4.0

See Also IsisSpaNew, IsisAppDelete.

IsisSpaFdt

#include<isisdll.h>

Long
FAR PASCAL

IsisSpaFdt(long *handle,*
*char *fdtfile*)

The **IsisSpaFdt** function specifies a specifies a Field Definition Table (fdt) file name.

Parameters

handle

Identifies the Isis Space.

fdtname

Specifies a field definition table file name file name or a string describing it. If it is a file name, it must be preceded by @. The specification of the extension is not required.

Return Value Possible values are:

- ZERO Success.
- ERR_FILEREAD File read error.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARFILNSIZ Invalid file name size.
- ERR_PARNULLPNT NULL pointer.

Comments This function changes the tag numbers of the IsisRecDump output by their field names. To return to the usual output an empty fdtfile parameter should be used.
Note that IsisRecUpdate function only works with tag number description of the record.

Example

```
IsisSpaFdt(H, "@"fdtfile);
IsisRecRead(H, 0, 1);
IsisRecDump(H, 0, area, areasize);
```

Unchanged since (version) β 7.0

See Also IsisSpaNew, IsisRecDump, IsisRecUpdate.

IsisSpaFst

#include<isisdll.h>

Long
FAR PASCAL

IsisSpaFst(*long handle,*
*char *fstfile)*

The **IsisSpaFst** function specifies an Isis Space Field Select Table (fst) file name.

Parameters

handle

Identifies the Isis Space.

fstfile

Specifies a field select table file name file name or a string describing it. If it is a file name, it must be preceded by @. The specification of the extension is not required.

Return Value Possible values are:

- ZERO Success.
- ERR_FILEFST File does not exist (fst).
- ERR_PARFILNSIZ Invalid file name size.

Comments This function verifies if the *fstname*.fst file exists.

Example

```
IsisSpaFst(H, "@c:\\bases\\lilacs\\lilacs");
```

Unchanged since (version) β 7.0

See Also IsisSpaNew, IsisSpaDb, IsisRecIfUpdate, IsisLnkIfLoad, IsisLnkIfLoadEx.

IsisSpaGf

#include<isisdll.h>

Long
FAR PASCAL

IsisSpaGf(*long handle,*

*char *gizmofile)*

The **IsisSpaGf** function specifies a database with patterns for changing sequences of characters in record fields.

Parameters

hadle

Identifies the Isis Space.

gizmo

Specifies the gizmo database name. The specification of the extension is not required.

Return Value

Possible values are:

- ZERO Success.
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.

Comments

This function verifies if the *gizmoname*.mst and gizmoname.xrf files exist.

Example

```
IsisSpaGf(H, "c:\\bases\\cds\\cdsgiz");
```

Unchanged since (version)

β 4.0

See Also

IsisSpaNew, IsisSpaDb.

IsisSpaHeaderMap

#include<isisdll.h>
#include<isis001.h>

long
FAR PASCAL

IsisSpaHeaderMap(*long handle,*
*char *sarea)*

The **IsisSpaHeaderMap** function copies the IsisSpaHeader structure into an application program area.

Parameters

handle

Identifies the Isis Space.

sarea

Specifies the pre-allocated area where the data will be copied.

Return Value

The return value is the Isis Space file status.

Comments

An IsisSpaHeader structure should be used to avoid memory corruption.

Example

```
IsisSpaHeader SpaHeader;

IsisSpaHeaderMap (H, TOCHAR (SpaHeader));
printf ("Space name: %s", SpaHeader.name);
```

Unchanged since (version) β 4.0

See Also IsisSpaNew.

IsisSpaIf

#include<isisdll.h>

long
FAR PASCAL

IsisSpaIf(*long handle,*
*char *iname*)

The **IsisSpaIf** function specifies the inverted file name for an Isis Space.

Parameters

handle

Identifies the Isis Space.

iname

Specifies the inverted file name. File extension is not required.

Return Value Possible values are:

- ZERO Success.
- ERR_FILEINVERT File does not exist (inverted).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARFILNSIZ Invalid file name size.

Comments The presence of all the required files (*.cnt, *.n01, *.n02, *.l01, *.l02, *.ifp) is verified. If any of these is missing ERR_FILEINVERT is returned.

Example

```
r = IsisSpaIf(H, "c:\\isis\\data\\cds");
```

Unchanged since (version) β 4.0

See Also IsisSpaNew, IsisSpaIfCreate, IsisSpaDb.

IsisSpaIfCreate

#include<isisdll.h>

long
FAR PASCAL

IsisSpaIfCreate(*long handle)*

The **IsisSpaIfCreate** function creates a new inverted file if it does not already exist otherwise the inverted file is re-initialized.

Parameters *handle*

Identifies the Isis Space.

Return Value Possible values are:

- ZERO Success.
- ERR_FILEINVERT File does not exist (inverted).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARFILNSIZ Invalid file name size.

Comments The inverted file name is specified by the IsisSpaIf function, otherwise ERR_FILEINVERT is returned.

Example

```
IsisSpaIf (H, "c:\\isis\\data\\cds");
IsisSpaIfCreate(H);
```

Unchanged since (version) β 4.0

See Also IsisSpaNew, IsisSpaIf, IsisSpaDb.

IsisSpaIsoDelim

#include<isisdll.h>

long
FAR PASCAL

IsisSpaIsoDelim(*long handle,*
*char *recdelim,*
*char *fielddelim)*

The **IsisSpaIsoDelim** function specifies the record and field delimiters for an ISO 2709 export file.

Parameters *handle*

Identifies the Isis Space.

recdelim

Specifies the record delimiter.

fielddelim

Specifies the field delimiter.

Return Value Possible values are:

- ZERO Success.
- ERR_PARNULLSTR String with zero size.

Comments The default value for record and field delimiters is #

Example

```
IsisSpaIsoOut (H, "c:\\isis\\data\\cds");
IsisSpaIsoDelim (H, "$", "&");
```

Unchanged since (version) β 4.0

See Also IsisSpaIsoOut, IsisRecIsoWrite.

IsisSpaIsoIn

#include<isisdll.h>

long
FAR PASCAL

IsoSpaIsoIn(*long handle,*
*char *isiname*)

The **IsoSpaIsoIn** function specifies an ISO 2709 import file name for an Isis Space.

Parameters

handle

Identifies the Isis Space.

isiname

Specifies the ISO 2709 import file name. The specification of the extension is not required.

Return Value Possible values are:

- ZERO Success.
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.

Comments When the ISO 2709 import file and the ISO 2709 export file are both specified in an Isis Space, they must have different names.

Example

```
IsisSpaIsoIn (H, "c:\\isis\\data\\cds");
```

Unchanged since (version) β 4.0

See Also IsisSpaIsoOut.

IsisSpaIsoOut

#include<isisdll.h>

long
FAR PASCAL

IsisSpaIsoOut(*long handle,*
*char *isoname*)

The **IsisSpaIsoOut** function specifies an ISO 2709 export file name for an Isis Space. If the file does not exist, it is initialized.

Parameters

handle

Identifies the Isis Space.

isoname

Specifies the ISO 2709 export file name. The specification of the extension is not required.

Return Value

Possible values are:

- ZERO Success.
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error..

Comments

When the ISO 2709 import file and the ISO 2709 export file are both specified in an Isis Space, they must have different names.

Example

```
IsisSpaIsoOut (H, "c:\\isis\\data\\backup");
```

Unchanged since (version)

β 4.0

See Also

IsisSpaIsoOutCreate, IsisSpaIsoIn.

IsisSpaIsoOutCreate

#include<isisdll.h>

long
FAR PASCAL
IsisSpaIsoOutCreate(*long handle)*

The **IsisSpaIsoOutCreate** function creates a new ISO 2709 export file if it does not exist otherwise, initializes it.

Parameters

handle

Identifies the Isis Space.

Return Value

Possible values are:

- ZERO Success.
- ERR_FILEISO File does not exist (ISO).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.

Comments

The ISO 2709 export file name is specified by IsisSpaIsoOut function.

Example

```
IsisSpaIsoOut(H, “c:\\isis\\data\\isoout”);
IsisSpaIsoOutCreate (H);
```

Unchanged since (version)

β 4.0

See Also

IsisSpaIsoOut.

IsisSpaMf

#include<isisdll.h>

long
FAR PASCAL

IsisSpaMf(long *handle,*
*char *mname*)

The **IsisSpaMf** function specifies an Isis Space master file name.

Parameters

handle

Identifies the Isis Space.

mname

Specifies the master file name. The specification of the extension is not required.

Return Value

Possible values are:

- ZERO Success.
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARFILNSIZ Invalid file name size.

Comments

This function verifies the presence of *mname*.mst and *mname.*xfr files. If either or both are missing ERR_FILEMASTER is returned.

Example

```
long A, H;

A = IsisAppNew();
H = IsisSpaNew(A);
IsisSpaMf(H, "c:\\bases\\lilacs\\lilacs");
```

Unchanged since (version)

β 4.0

See Also

IsisSpaNew, IsisSpaMfCreate, IsisSpaDb.

IsisSpaMfCreate

#include<isisdll.h>

Long
FAR PASCAL

IsisSpaMfCreate(*long handle)*

The **IsisSpaMfCreate** function creates a new master file or initializes an existing master file. The creation or initialization applies to files with extension .mst and .xrf.

Parameters

Handle

Identifies the Isis Space.

Return Value

Possible values are:

- ZERO Success.
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARFILNSIZ Invalid file name size.

Comments

The master file name should be specified by the IsisSpaMf function, otherwise ERR_FILEMASTER is returned.

Example

```
long A, H;

A = IsisAppNew();
H = IsisSpaNew(A);

IsisSpaMf(H, "c:\\bases\\lilacs\\lilacs");
IsisSpaMfCreate(H);
```

Unchanged since (version)

β 4.0

See Also

IsisSpaNew, IsisSpaMf, IsisSpaDb.

IsisSpaMfUnlockForce

#include<isisdll.h>

Long
FAR PASCAL

IsisSpaMfUnlockForce(*long handle)*

The **IsisSpaMfUnlockForce** function resets the data entry lock and the exclusive write lock flags of the data base.

Parameters

Handle

Identifies the Isis Space.

Return Value

Possible values are:

- ZERO Success.
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_RECLOCKED Record locked.

Comments

Data entry lock and exclusive write lock flags can be verified with the IsisRecControlMap function. This function does not unlock the locked records of the data base, for this use IsisRecUnlock or IsisRecUnlockForce functions.

Example

```
IsisSpaMf(H, "c:\\bases\\lilacs\\lilacs");
IsisSpaMfUnlockForce(H);
```

Unchanged since (version) β 5.0

See Also IsisSpaMf, IsisSpaDb, IsisRecControlMap, IsisRecUnlock, IsisRecUnlockForce.

IsisSpaName

#include<isisdll.h>

long
FAR PASCAL

IsisSpaName *(long handle,*
*char *sname)*

The **IsisSpaName** function gives a name to an Isis Space.

Parameters

handle

Identifies the Isis Space.

sname

Specifies the Isis Space name

Return Value Possible values are:

- ZERO Success.
- ERR_PARNULLSTR String with zero size.

Comments The Isis Space name can be retrieved by the function IsisSpaHeaderMap.

Example

```
IsisSpaName (H, "My Isis Space");
```

Unchanged since (version) β 4.0

See Also IsisSpaNew, IsisSpaHeaderMap.

IsisSpaNew

#include<isisdll.h>

long
FAR PASCAL

IsisSpaNew*(long apphandle)*

The **IsisSpaNew** function creates a new Isis Space.

Parameters

apphandle

Authorization to create a new Isis Space.

Return Value

Possible values are:

- handle of Isis Space, if positive
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARAPPHAND Invalid application handle.

Comments

The *apphandle* parameter is obtained with a call to IsisAppNew function.

Example

```
long A, H;

A = IsisAppNew ();
H = IsisSpaNew (A);

IsisSpaDb (H, "c:\\isis\\data\\cds");
```

Unchanged since (version)

β 4.0

See Also

IsisAppNew, IsisSpaDelete.

IsisSpaPft

#include<isisdll.h>

long
FAR PASCAL

IsisSpaPft(long *handle,*
*char *pftname*)

The **IsisSpaPft** function specifies an Isis Space format.

Parameters

handle

Identifies the Isis Space.

pftname

Specifies the format file name or a string describing the format. If it is a file name, it must be preceded by @. The specification of the extension is not required.

Return Value

Possible values are:

- ZERO Success.
- if > 0 The position of syntax error (first pos = 1).
- if < 0 The error code which can be:
  - ERR_FILEPFT File does not exist (pft).
  - ERR_FILEREAD File read error.
  - ERR_LLCISISETRAP Cisis Low Level Error Trap.
  - ERR_MEMALLOCAT Memory Allocation Error.
  - ERR_PARFILNSIZ Invalid file name size.
  - ERR_PARFMTSYNT Syntax Error (format)

Comments This format specification should be used by the IsisRecFormat function. If an empty or null pftname parameter is specified, the IsisRecFormat output will be the dump of the record.

Example
```
r = IsisSpaPft(H, "@c:\\bases\\lilacs\\lilacs");
r = IsisSpaPft(H, "mfn/v24/");
```

Unchanged since (version) β 7.0

See Also IsisSpaNew, IsisSpaDb, IsisRecFormat, IsisRecFormatEx.

IsisSpaRecDelim

#include<isisdll.h>

long
FAR PASCAL

IsisSpaRecDelim(*long handle,*
*char *begindelim,*
*char *enddelim)*

The **IsisSpaRecDelim** function specifies the start and end tag delimiters to be used by the functions IsisRecDump and IsisRecUpdate.

Parameters

handle

Identifies the Isis Space.

begindelim

Specifies the start tag delimiter.

enddelim

Specifies the end tag delimiter.

Return Value Possible values are:

- ZERO Success.
- ERR_PARNULLSTR String with zero size.

Comments The default values for the delimiters are ‘<‘ and ‘>‘.

Example
```
IsisSpaRecDelim (H, "{", "}" );
```

Unchanged since (version) β 4.0

See Also IsisSpaNew, IsisRecDump, IsisRecUpdate.

IsisSpaRecShelves

#include<isisdll.h>

long
FAR PASCAL

IsisSpaRecShelves(*long handle,*
long mmax)

The **IsisSpaRecShelves** function specifies the maximum number of master file record shelves available in an Isis Space, i.e., the maximum number of master file records that can be simultaneously loaded into a space.

Parameters

handle

Identifies the Isis Space.

mmax

Specifies the number of record shelves (mmax >= 0).

Return Value

Possible values are:

- ZERO Success.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments

The default number of record shelves when the Isis Space is created is one. The table bellow describes the action taken by the function according to the value of *mmax*

mmax	*Action*
zero	deletes all available shelves.
less than the current	deletes all the exceeding shelves.
equal to the current	clears the contents of all shelves.
greater than the current	preserves the contents of the old shelves.

Example

```
H = IsisSpaNew (A);

IsisSpaRecShelves (H, 3);
```

Unchanged since (version)

β 4.0

See Also

IsisSpaNew, IsisSpaTrmShelves, IsisRecShelfSize.

IsisSpaStw

#include<isisdll.h>

long
FAR PASCAL

IsisSpaStw(long *handle,*
*char *stwname*)

The **IsisSpaStw** function specifies an Isis Space stopword file name (stw).

Parameters

handle

Identifies the Isis Space.

stwname

Specifies the stopword file name or a string describing them. If it is a file name, it must be preceded by @. The specification of the extension is not required.

Return Value Possible values are:

- ZERO Success.
- ERR_FILESTW File does not exist (stw).
- ERR_PARFILNSIZ Invalid file name size.

Comments This function verifies the presence of *stwname*.stw file. The stopword file is used by IsisRecIfUpdate and IsisRecLnk functions for indexing techniques 4 and 8 (word indexing).

Example

```
r = IsisSpaStw(H, "@c:\\bases\\lilacs\\lilacs");
```

Unchanged since (version) β 7.0

See Also IsisSpaNew, IsisSpaDb, IsisSpaFst.

IsisSpaTrmShelves

#include<isisdll.h>

long
FAR PASCAL

IsisSpaTrmShelves(*long handle,*
long tmax)

The **IsisSpaTrmShelves** function specifies the maximum number of term shelves available in an Isis Space, i.e., the maximum number of terms that can be simultaneously loaded into a space.

Parameters

handle

Identifies the Isis Space.

tmax

Specifies the number of term shelves (tmax >= 0).

Return Value Possible values are:

- ZERO Success.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments The default number of term shelves when the Isis Space is created is one. The table bellow describes the action taken by the function according to the value of *tmax*

tmax *Action*

zero	deletes all available shelves.
less than the current	deletes all the exceeding shelves.
equal to the current	clears the contents of all shelves.
greater than the current	preserves the contents of the old shelves.

Example

```
H = IsisSpaNew (A);

IsisSpaTrmShelves (H, 2);
```

Unchanged since (version) β 4.0

See Also IsisSpaNew, IsisSpaRecShelves, IsisTrmShelfSize.

IsisSrcHeaderMap

#include<isisdll.h>
#include <isis001.h>

long
FAR PASCAL

IsisSrcHeaderMap(*long apphandle,*
long tsfnum,
long searchnum,
*char *sstrup*)

The **IsisSrcHeaderMap** function copies the search header control data to an application program area.

Parameters

apphandle

Identifies an Isis Application.

tsfnum

Specifies the number of the temporary search log file containing the desired search results (tsfnum >= 0).

searchnum

Specifies the search number . If this parameter is zero, the last search submitted will be retrieved (searchnum >= 0).

sstrup

Specifies the pre-allocated area where the data will be copied.

Return Value Possible values are:

- number of records retrieved
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments The IsisSrcHeader structure should be used to avoid memory corruption.

Example

```
IsisSrcHeader  Hdrstru;
IsisSrcMfn     MfnStru;
```

```
long           Cont;
 ...

IsisSrcHeaderMap (A, 0, 15, TOCHAR(Hdrstru));

for (Cont=0; Cont < Hdrstru.recs ; Cont++)
{
  IsisSrcMfnMap (A, 0, 0, Cont, TOCHAR(MfnStru));
  printf ("mfn:%ld \n", MfnStru.mfn)
}
```

Unchanged since (version) β 7.0

See Also IsisSrcSearch, IsisSrcMfnMap.

IsisSrcLogFileFlush

#include<isisdll.h>

long
FAR PASCAL

IsisSrcLogFileFlush(*long apphandle,*
long lognumber)

The **IsisSrcLogFileFlush** function deletes the contents of a search log file.

Parameters

apphandle

Identifies an Isis Application.

lognumber

Specifies the number of the search log file (*lognumber*).

Return Value Possible values are:

- ZERO Success.
- ERR_FILEDELETE File delete error.
- ERR_PAROUTRANG Parameter out of range.

Comments

Example

```
IsisSrcLogFileFlush (A, 3);
```

Unchanged since (version) β 7.0

See Also IsisSrcSearch, IsisSrcLogFileUse, IsisSrcLogFileSave.

IsisSrcLogFileSave

#include<isisdll.h>

long
FAR PASCAL

IsisSrcLogFileSave(*long apphandle,*
long lognumber,
*char *filename*)

The **IsisSrcLogFileSave** function save permanently the contents of a search log file in a given moment.

Parameters

apphandle

Identifies an Isis Application.

lognumber

Specifies the number of the search log file used by the search algorithm containing the desired search results (*lognumber* >= 0).

filename

Specifies the log file name.

Return Value

Possible values are:

- ZERO Success.
- ERR_FILEOPEN File open error.
- ERR_FILERENAME File rename error.
- ERR_PARFILNSIZ Invalid file name size.
- ERR_PAROUTRANG Parameter out of range.

Comments

After the execution of an application, all temporary search files will be deleted.
When a search log file is specified but no search is done, the file will be created but it's size will be *zero*.

Example

```
IsisSrcSearch (H, 3, Expression, TOCHAR(Hdrstru));

IsisSrcLogFileSave (H, 3, "c:\\app\\srclog2.log");
```

Unchanged since (version)

β 7.0

See Also

IsisSrcSeach, IsisSrcLogFileFlush, IsisSrcLogFileUse.

IsisSrcLogFileUse

#include<isisdll.h>

long
FAR PASCAL

IsisSrcLogFileUse(*long apphandle,*
*char *filename,*
long lognumber)

The **IsisSrcLogFileUse** function assigns to a search log file the searches previously saved by means of IsisSrcLogFileSave function.

Parameters

apphandle

Identifies an Isis Application.

filename

Specifies the file name.

lognumber

Specifies the number of the search log file to be associated with the file (*lognumber* >= 0).

Return Value

Possible values are:

- ZERO — Success.
- ERR_FILEOPEN — File open error.
- ERR_FILERENAME — File rename error.
- ERR_PARFILNSIZ — Invalid file name size.
- ERR_PAROUTRANG — Parameter out of range.

Comments

This function allows cross application search using the IsisSrcLogFileSave function.

Example

```
IsisSrcLogFileUse (A, "c:\\app\\srclog2.log", 3);

IsisSrcSearch (H, 3, Expression, TOCHAR(Hdrstru));
```

Unchanged since (version)

β 7.0

See Also

IsisSrcSearch, IsisSrcLogFileFlush, IsisSrcLogFileSave.

IsisSrcMfnMap

#include<isisdll.h>
#include<isis001.h>

long
FAR PASCAL

IsisSrcMfnMap (*long apphandle,*
long tsfnum,
long serchnum,
long firstpos,
long lastpos,
*char *mfnareap*)

The **IsisSrcMfnMap** function copies a subset of mfns from a search hit list and moves it into an application program area.

Parameters

apphandle

Identifies an Isis Application.

tsfnum

Specifies the number of the temporary search log file used by the search algorithm containing the desired search results (*tsfnum* >= 0).

searchnum

Specifies the search number. If this parameter is zero, the last search submitted will be retrieved (*searchnum* >= 0).

firstpos

Initial range position in the list of retrieved Mfns (*firstpos* > 0).

lastpos

Last range position in the list of retrieved Mfns (*lastpos* >= *firstpos*).

mfnareap

Specifies an array of IsisSrcMfn structures of appropriate size where the data will be copied

Return Value

Possible values are:

- number of Mfns
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments

The retrieved Mfns list starts with one. Uses IsisSrcHeader.recs to loop through the MFNs list. Note that repeated Mfns in the Hit list are ignored by IsisSrcMfnMap function. If the *lastpos* parameter is greater than the actual number of retrieved Mfns the later is used.

Example

```
IsisSrcHeader  Hdrstru;
IsisSrcMfn    *Mfnstrup;
long           Cont;
 ...

IsisSrcSearch (H, 0, Expression, TOCHAR(Hdrstru));
Mfnstrup = new IsisSrcMfn[Hdrstru.recs];
IsisSrcMfnMap (A, 0, 0, 1, Hdrstru.recs,
                                        (char *)Mfnstrup);

for (Cont=0; Cont < Hdrstru.recs ; Cont++)
  printf ("mfn:%ld \n", Mfnstrup[Cont].mfn);

delete[] Mfnstrup;
```

Unchanged since (version)

β 4.0

See Also

IsisSrcSearch, IsisSrcSearchEx, IsisSrcHeaderMap.

IsisSrcRegExpMap

#include<isisdll.h>
#include<isis001.h>

long
FAR PASCAL
IsisSrcRegExpMap(*long handle,*
*char *express,*
long beginmfn,
long endmfn,
*char *areap,*
long areasize)

The **IsisSrcRegExpMap** function executes a regular expression search over an Isis Space master file and copies basic data on the result to an application program area.

Parameters

handle

Identifies the Isis Space.

express

Specifies the search regular expression using grep like regular expression language.

beginmfn

Specifies the number of the initial mfn of a range in which the search will be applied (*beginmfn* >= 0).

endmfn

Specifies the number of the last mfn of a range in which the search will be applied (*beginmfn* >= 0).

areap

Specifies the area where the data will be copied.

areasize

Specifies the size of area.

Return Value

Possible values are:

- the search number
- ERR_FILEINVERT File does not exist (inverted).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARNULLPNT NULL pointer.
- ERR_PARNULLSTR String with zero size.
- ERR_PAROUTRANG Parameter out of range.
- ERR_PARSRCSYNT Syntax Error (search).

Comments

A regular expression is one or more occurrences of: One or more characters optionally enclosed in quotes. The following symbols are treated specially:

^ start of line	$ end of line
. any character	\ quote next character
* match zero or more	+ match one or more

[aeiou0-9] match a, e, i, o, u, and 0 thru 9
[^aeiou0-9] match anything but a, e, i, o, u, and 0 thru 9

The IsisSrcRegExp structure should be used as *areap* parameter to avoid memory corruption.

Example

```
IsisSrcRegExp RegExpStru[100];

IsisSrcRegExpMap(H, "wat+", 1, 30, TOCHAR(RegExpStru), 100);
```

Unchanged since (version)

β 7.0

See Also

IsisSpaMf, IsisSrcSearch, IsisSrcSearchEx.

IsisSrcSearch

#include<isisdll.h>
#include<isis001.h>

long
FAR PASCAL

IsisSrcSearch(*long handle,*
long tsfnum,
*char *express,*
*char *areap*)

The **IsisSrcSearch** function executes a search expression over an Isis Space inverted file and copies basic data on the result to an application program area.

Parameters

handle

Identifies the Isis Space.

tsfnum

Specifies the number of the temporary search log file to be used by the search algorithm to store the search results (*tsfnum* >= 0).

express

Specifies the search expression using the CDS/ISIS search language.

areap

Specifies the area where the data will be copied.

Return Value

Possible values are:

- the search number
- ERR_FILEINVERT File does not exist (inverted).
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARNULLPNT NULL pointer.
- ERR_PARNULLSTR String with zero size.
- ERR_PAROUTRANG Parameter out of range.
- ERR_PARSRCSYNT Syntax Error (search).

Comments

The search expression combines terms with Boolean operators such as:

Or	+	A+B
And	*	A*B
And Not	^	A^B

The IsisSrcHeader structure should be used as *areap* parameter to avoid memory corruption.

Example

```
char Expression[SRC_EXPR_LENGTH];
IsisSrcHeader   Hdrstru;

IsisSpaIf (H, Dbname);

IsisSrcSearch (H, 0, Expression, TOCHAR(Hdrstru));
```

Unchanged since (version) β 4.0

See Also IsisSpaMf, IsisSpaIf, IsisSpaDb, IsisSrcLogFileFlush, IsisSrcLogFileUse, IsisSrcHeaderMap, IsisSrcMfnMap, IsisSrcRegExpMap, IsisSrcSearchEx.

IsisSrcSearchEx

#include<isisdll.h>
#include<isis001.h>

long
FAR PASCAL

IsisSrcSearchEx(*long handle,*
long tsfnum,
*char *itable,*
*char *express,*
*char *areap*)

The **IsisSrcSearchEx** function executes a search expression over one or more Isis Space inverted files associated with prefixes in a search conversion table and copies basic data on the result to an application program area.

Parameters

handle

Identifies the Isis Space.

tsfnum

Specifies the number of the temporary search log file to be used by the search algorithm to store the search results (*tsfnum* >= 0).

itable

Specifies the search conversion table. If it is a file name, it must be preceded by @. The specification of the extension is not required.

express

Specifies the search expression using the CDS/ISIS search language.

areap

Specifies the area where the data will be copied.

Return Value

Possible values are:

- the search number
- ERR_FILEMASTER File does not exist (master).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARNULLPNT NULL pointer.
- ERR_PARNULLSTR String with zero size.
- ERR_PAROUTRANG Parameter out of range.
- ERR_PARSRCSYNT Syntax Error (search).

Comments

The search expression combines prefixed terms with Boolean operators such as:

"Luziadas * Camoes"
"AUT Camoes"
"TIT Luziadas"
"AUT Shakespeare + TIT Luziadas + Camoes""[AUT] (Shakespeare + Camoes) * TI Luziadas"

The IsisSrcHeader structure should be used as *areap* parameter to avoid memory corruption.

Example

```
char Expression[SRC_EXPR_LENGTH];
IsisSrcHeader   Hdrstru;

IsisSrcSearchEx (H, 0, @Table, Expression,
                                        TOCHAR(Hdrstru));
```

Unchanged since (version) β 7.0

See Also IsisSpaMf, IsisSpaIf, IsisSpaDb, IsisSrcLogFileFlush, IsisSrcLogFileUse, IsisSrcHeaderMap, IsisSrcMfnMap, IsisSrcSearch.

IsisTrmMfnMap

#include<isisdll.h>
#include<isis001.h>

long
FAR PASCAL

IsisTrmMfnMap(*long handle,*
long index,
long firstpos,
long lastpos,
char mfnareap)

The **IsisTrmMfnMap** function copies a subset of mfns from a term posting list and moves it into an application program area. The posting corresponds to the term stored at shelf *index*.

Parameters

handle

Identifies the Isis Space.

index

Specifies the term shelf number.

firstpos

Initial range position in the list of retrieved Mfns (*firstpos* > 0).

lastpos

Last range position in the list of retrieved Mfns (*lastpos* >= *firstpos*).

mfnareap

Specifies an array of IsisTrmMfn structures of appropriate size where the data will be copied.

Return Value Possible values are:

- number of Mfns.
- ERR_FILEINVERT File does not exist (inverted).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PARNULLPNT NULL pointer.
- ERR_PAROUTRANG Parameter out of range.

Comments The retrieved Mfns list starts with one. Uses the total number of postings (IsisTrmReadMap) to loop through the MFNs list. Note that repeated Mfns in the Posting list are ignored by IsisTrmMfnMap function. If the *lastpos* parameter is greater than the actual number of retrieved Mfns the later is used.

Example

```
PostNum = IsisTrmReadMap(H, 0, trmread);
Mfnareap = new IsisTrmMfn[PostNum];
MfnNum = IsisTrmMfnMap (H, 0, 1, PostNum,
                                        (char *)MfnAreap);

for (Cont = 0; Cont < MfnNum; Cont++)
  printf ("mfn:%ld \n", Mfnareap[Cont].mfn);

PostNum = IsisTrmReadNext(H, 0, trmread);
```

Unchanged since (version) β 4.0

See Also IsisTrmReadMap, IsisTrmReadNext, IsisTrmReadPrevious, IsisTrmShelfSize.

IsisTrmPostingMap

#include<isisdll.h>
#include<isis001.h>

long
FAR PASCAL

IsisTrmPostingMap(*long handle,*
long index,
long firstpos,
long lastpos,
*char *parea*)

The **IsisTrmPostingMap** function copies a subset of postings into an application program area. The postings correspond to the term stored at shelf *index*.

Parameters

handle

Identifies the Isis Space.

index

Specifies the term shelf number.

firstpos

Initial range position in the list of retrieved posting elements (*firstpos* > 0).

lastpos
Last range position in the list of retrieved postings elements (*lastpos* >= *firstpos*).

parea
Specifies an array of IsisTrmPosting structures where the data will be copied.

Return Value

Possible values are:

- number of retrieved postings.
- ERR_FILEINVERT File does not exist (inverted).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.

Comments

The *posting_elm* area should be an IsisTrmPosting structure to avoid memory corruption.
To load posting number *n*, it is necessary to previously load all the preceding *n-1* postings. There is no direct access to a given postings.

Example

```
IsisTrmRead     TrmRead;
IsisTrmPosting *TrmPosting;
long            TotPost, Cont;

IsisSpaIf(H, "c:\\bases\\cds\\cds");

TrmRead.key[0] = '\0';
TotPost = IsisTrmReadMap(H, 0, TOCHAR(TrmRead));

TrmPosting = new IsisTrmPosting[TotPost];
if (TrmPosting = NULL)
 exit (EXIT_FAILURE);

IsisTrmPostingMap(H, 0, 1, TotPost, TrmPosting);

While (PostNum != ERR_TRMEOF)
{
 for (Cont = 0; Cont < TotPost; Cont++)
 {
printf ("posting:%ld mfn:%ld tag:%ld occ:%ld cnt:%ld",
   TrmPosting[Cont].posting, TrmPosting[Cont].mfn,
TrmPosting[Cont].tag,   TrmPosting[Cont].occ,
TrmPosting[Cont].cnt);
 }

 TotPost = IsisTrmReadNext(H, 0, TOCHAR(TrmRead));
}

delete[] TrmPosting;
```

Unchanged since (version)

β 5.0

See Also

IsisTrmReadMap, IsisTrmShelfSize.

IsisTrmReadMap

#include<isisdll.h>
#include<isis001.h>

long
FAR PASCAL

IsisTrmReadMap(*long handle,*
long index,
*char *key*);

The **IsisTrmReadMap** function loads an inverted file term into a shelf and returns the number of postings or an error condition.

Parameters

handle

Identifies the Isis Space.

index

Specifies the term shelf number.

key

Specifies the IsisTrmRead structure with the inverted file term to be loaded.

Return Value

Possible values are:

- total number of postings
- ERR_FILEINVERT File does not exist (inverted).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.
- ERR_TRMEOF Term eof: found eof in data base.
- ERR_TRMNEXT Term next: key not found.

Comments

Unless the shelf *index* had not been previously allocated with zero bytes (IsisTrmShelfSize), this function loads the corresponding posting header segment.

IsisTrmReadMap always converts key to uppercase.

Example

```
IsisTrmRead TrmRead;
long PostNum;
long  Rc;

Rc = IsisSpaIf(H, "c:\\bases\\cds\\cds");

TrmRead.key[0] = '\0';
PostNum = IsisTrmReadMap(H, 0, TOCHAR(TrmRead));
```

Unchanged since (version)

β 5.0

See Also

IsisSpaIf, IsisSpaDb, IsisTrmMfnMap, IsisTrmReadNext, IsisTrmReadPrevious, IsisTrmMfnMap, IsisTrmPostingMap.

IsisTrmReadNext

#include<isisdll.h>
#include<isis001.h>

long
FAR PASCAL

IsisTrmReadNext(*long handle,*
long index,
*char *key*)

The **IsisTrmReadNext** function loads the next inverted file term and returns the number of postings or an error condition.

Parameters

handle

Identifies the Isis Space.

index

Specifies the term shelf number.

key

Specifies the IsisTrmRead structure where the next inverted file term will be copied.

Return Value

Possible values are:

- total number of postings
- ERR_FILEINVERT File does not exist (inverted).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.
- ERR_TRMEOF Term eof: found eof in data base.

Comments

Unless the shelf *index* had not been previously allocated with zero bytes (IsisTrmShelfSize), this function loads the corresponding posting header segment.
A call to IsisTrmReadNext must be preceded by a call to IsisTrmReadMap, IsisTrmReadPrevious or another IsisTrmReadNext.

Example

```
IsisTrmRead TrmRead;
long Rc;

r = IsisSpaIf(H, "c:\\bases\\cds\\cds");

TrmRead.key[0] = '\0';
Rc = IsisTrmReadMap(H, 0, TOCHAR(TrmRead));
while (Rc != ERR_TRMEOF)
 Rc = IsisTrmReadNext(H, 0, TOCHAR(TrmRead));
```

Unchanged since (version)

β 5.0

See Also

IsisTrmReadMap, IsisTrmReadPrevious, IsisTrmPostingMap, IsisTrmMfnMap.

IsisTrmReadPrevious

#include<isisdll.h>

#include<isis001.h>

long
FAR PASCAL

IsisTrmReadPrevious(*long handle,*
long index,
*char *prefix,*
*char *key*)

The **IsisTrmReadPrevious** function loads the previous inverted file term and returns the number of postings or an error condition.

Parameters

handle

Identifies the Isis Space.

index

Specifies the term shelf number.

prefix

Specifies the lower limit for the previous key.

key

Specifies the IsisTrmRead structure where the previous inverted file term will be copied.

Return Value

Possible values are:

- total number of postings
- ERR_FILEINVERT File does not exist (inverted).
- ERR_LLCISISETRAP Cisis Low Level Error Trap.
- ERR_MEMALLOCAT Memory Allocation Error.
- ERR_PAROUTRANG Parameter out of range.
- ERR_TRMEOF Term eof: found eof in data base.

Comments

Unless the shelf *index* had not been previously allocated with zero bytes, this function loads the corresponding posting header segment.
A call to IsisTrmReadPrevious must be preceded by a call to IsisTrmReadMap, IsisTrmReadNext or another IsisTrmReadPrevious.

Example

```
IsisTrmRead TrmRead;
long Rc;

r = IsisSpaIf(H, "c:\\bases\\cds\\cds");

TrmRead.key[0] = '\0';
Rc = IsisTrmReadMap(H, 0, TOCHAR(TrmRead));
while (Rc >= 0)
 Rc = IsisTrmReadPrevious(H, 0, "ab", TOCHAR(TrmRead));
```

Unchanged since (version)

β 5.0

See Also

IsisTrmReadMap, IsisTrmReadNext, IsisTrmPostingMap, IsisTrmMfnMap.

IsisTrmShelfSize

#include<isisdll.h>

long
FAR PASCAL

IsisTrmShelfSize(*long handle,*
long index,
long mem)

The **IsisTrmShelfSize** function changes the size of an inverted file shelf.

Parameters

handle

Identifies the Isis Space.

index

Specifies the term shelf number.

mem

Specifies the number of bytes to be allocated (0 or 512).

Return Value

Possible values are:

- ZERO — Success.
- ERR_FILEINVERT — File does not exist (inverted).
- ERR_LLCISISETRAP — Cisis Low Level Error Trap.
- ERR_MEMALLOCAT — Memory Allocation Error.
- ERR_PAROUTRANG — Parameter out of range.

Comments

The default value is 512. If it is *zero*, postings are not loaded, increasing the execution speed but preventing access to the postings of the term (IsisTrmPostingMap, IsisTrmMfnMap).

Example

```
IsisSpaTrmShelfSize (H, 0, 0);
```

Unchanged since (version)

β 4.0

See Also

IsisSpaTrmShelves, IsisRecShelfSize, IsisTrmPostingMap, IsisTrmMfnMap.

Annex 1 - Data Definitions and Structures

1. ISIS_DLL Global Constants.

1.1. Debug Flags.

```
const SHOW_NEVER          =   0;
const SHOW_FATAL          =   1;
const SHOW_ALWAYS=    3;

const EXIT_NEVER          =   0;
const EXIT_FATAL          =  16;
const EXIT_ALWAYS         =  48;

const DEBUG_VERY_LIGHT    =  SHOW_NEVER    | EXIT_NEVER;
const DEBUG_LIGHT         =  SHOW_FATAL    | EXIT_FATAL;
const DEBUG_HARD          =  SHOW_ALWAYS   | EXIT_FATAL;
const DEBUG_VERY_HARD     =  SHOW_ALWAYS   | EXIT_ALWAYS;
```

1.2. General Constants.

```
const KEY_LENGTH          =    30;          /* Term key length. */
const IFBSIZE             =   512;          /* Default term shelf size. */
const SRC_EXPR_LENGTH     =   512;          /* Maximum search expr length.*/
/* const SRC_EXPR_LENGTH  =   254;          /* DELPHI max search expr length.*/
const MAXPATHLEN          =    63;          /* Maximum file path length. */
const MAXMFRL             = 30000;          /* Maximum record length. */
```

2. ISIS_DLL Structures.

2.1. C/C++ Structures.

2.1.1. Record Structures.

```
struct IsisRecControl
{
  long          ctlmfn;                     /* gdb ctlmfn. */
  long          nxtmfn;                     /* gdb nxtmfn. */
  long          nxtmfb;                     /* gdb nxtmfb. */
  long          nxtmfp;                     /* gdb nxtmfp - offset. */
```

```
  long          mftype;                 /* gdb mftypedef. */
  long          reccnt;                 /* gdb reccnt. */
  long          mfcxx1;                 /* gdb mfcxx1. */
  long          mfcxx2;                 /* gdb mfcxx2 - MULTI: Data entry lock. */
  long          mfcxx3;                 /* gdb mfcxx3 - MULTI: Exclusive write  lock. */
};

struct IsisRecDir
{
  long           tag;                   /* field tag entry. */
  long           pos;                   /* field position. */
  long           len;                   /* field length entry. */
};

struct IsisRecLeader
{
  long          mfn;                    /* gdb mfn. */
  long          mfrl;                   /* gdb mfrl - MULTI: record being updated. */
  long          mfbwb;                  /* gdb mfbwb. */
  long          mfbwp;                  /* gdb mfbwp - offset. */
  long          base;                   /* gdb base (MSNVSPLT). */
  long          nvf;                    /* gdb nvf. */
  long          status;                 /* gdb status. */
};
```

2.1.2. Isis Space Structures.

```
struct IsisSpaHeader
{
  long          handle;                 /* pointer ISIS_SPACE. */
  char          name[MAXPATHLEN+1];     /* ISIS_SPACE name. */
  char          cipar[MAXPATHLEN+1];    /* cipar file name. */
  char          mf[MAXPATHLEN+1];       /* master file name. */
  char          ifi[MAXPATHLEN+1];      /* inverted file name. */
   char         fst[MAXPATHLEN+1];      /* fst file name. */
  char          pft[MAXPATHLEN+1];      /* pft file name. */
  char          fmt[MAXPATHLEN+1];      /* fmt file name. */
  char          stw[MAXPATHLEN+1];      /* stw file name. */
  char          fdt[MAXPATHLEN+1];      /* fdt file name. */
  char          isoin[MAXPATHLEN+1];    /* import iso file name.*/
  char          isoout[MAXPATHLEN+1];   /* export iso file name.*/
   char         giz[MAXPATHLEN+1];     /* gizmo file name. */
  char          dec[MAXPATHLEN+1];     /* decode file name. */
  long          rec;                    /* number of RECSTRU shelves. */
  long          trm;                    /* number of TRMSTRU shelves. */
  long          filestatus;             /* file status - bit mask. */
};
```

2.1.3. Search Structures.

```
struct IsisSrcHeader
{
  long          number;                                 /* search number (start in 1). */
  long          recs;                                   /* total records retrieved. */
  char          dbname[MAXPATHLEN+1];                   /* data base name. */
  char          booleanexpr[SRC_EXPR_LENGTH+1]; /* search expression. */
};


struct IsisSrcMfn
{
  long          mfn;                                    /* hit mfn component. */
};


struct IsisSrcRegExp
{
 long    mfn;                                          /* mfn where the expression was found */
 long    tag;                                          /* tag where the expression was found */
 char    expr[SRC_EXPR_LENGTH + 1];                    /* retrieved expression */
};
```

2.1.4. Term Structures.

```
struct IsisTrmMfn
{
  long          mfn;                                    /* posting mfn component. */
};


struct IsisTrmPosting
{
  long          posting;                          /* current posting order. */
  long          mfn;                                    /* current posting pmfn. */
  long          tag;                                    /* current posting ptag. */
  long          occ;                                    /* current posting pocc. */
  long          cnt;                                    /* current posting pcnt. */
};


struct IsisTrmRead
{
  char          key[KEY_LENGTH+1];                      /* term key.*/
};
```

2.2. Visual Basic Structures.

```
Global Const KEY_LENGTH1 = KEY_LENGTH + 1
Global Const SRC_EXPR_LENGTH1 = SRC_EXPR_LENGTH + 1
Global Const MAXPATHLEN1 = MAXPATHLEN + 1
```

2.2.1. Record Structures.

```
Type IsisRecControl

  ctlmfn        As Long                    'gdb ctlmfn.
  nxtmfn        As Long                    'gdb nxtmfn.
  nxtmfb        As Long                    'gdb nxtmfb.
  nxtmfp        As Long                    'gdb nxtmfp - offset.
  mftype        As Long                    'gdb mftype.
  reccnt        As Long                    'gdb reccnt.
  mfcxx1        As Long                    'gdb mfcxx1.
  mfcxx2        As Long                    'gdb mfcxx2 - MULTI: Data entry lock.
  mfcxx3        As Long                    'gdb mfcxx3 - MULTI: Exclusive write
lock.

End Type


Type IsisRecDir

  tag           As Long                    'field tag entry.
  pos           As Long                    'field position.
  len           As Long                    'field length entry.

End Type


Type IsisRecLeader

  mfn           As Long                        'gdb mfn.
  mfrl          As Long                        'gdb mfrl - MULTI: record being updated.
  mfbwbAs Long                         'gdb mfbwb.
  mfbwpAs Long                         'gdb mfbwp - offset.
  base          As Long                        'gdb base (MSNVSPLT).
  nvf           As Long                        'gdb nvf.
  status        As Long                        'gdb status.

End Type
```

2.2.2. Isis Space Structures.

```
Type IsisSpaHeader

  handle        As Long                    'pointer to ISIS_SPACE.
  name          As String * MAXPATHLEN1    'ISIS_SPACE name.
  cipar         As String * MAXPATHLEN1    'cipar file name.
  mf            As String * MAXPATHLEN1    'master file name.
  ifi           As String * MAXPATHLEN1    'inverted file name.
  isoin         As String * MAXPATHLEN1    'import iso file name.
  isoout        As String * MAXPATHLEN1    'export iso file name.
  rec           As Long                    'number of RECSTRU shelves.
  trm           As Long                    'number of TRMSTRU shelves.
  filestatus    As Long                    'file status - bit mask.

End Type
```

2.2.3. Search Structures.

```
Type IsisSrcHeader

  number       As Long                       'search number (start in 1).
  recs         As Long                       'total records retrieved.
  dbname       As String * MAXPATHLEN1       'data base name.
  booleanexpr  As String * SRC_EXPR_LENGTH1  'search expression.

End Type

Type IsisSrcMfn

  mfn          As Long                       'hit mfn component

End Type
```

2.2.4. Term Structures.

```
Type IsisTrmMfn

  mfn          As Long                       'hit mfn component

End Type

Type IsisTrmPosting

  posting      As Long                       'current posting order.
  mfn          As Long                       'current posting pmfn.
  tag          As Long                       'current posting ptag.
  occ          As Long                       'current posting pocc.
  cnt          As Long                       'current posting pcnt.

End Type

Type IsisTrmRead

  key          As String *KEY_LENGTH1        'term key.

End Type
```

2.3. Delphi Structures.

2.3.1. Record Structures.

```
Type IsisRecControl = record

  ctlmfn:      LongInt;                          {gdb ctlmfn.}
  nxtmfn:      LongInt;                          {gdb nxtmfn.}
  nxtmfb:      LongInt;                          {gdb nxtmfb.}
  nxtmfp:      LongInt;                          {gdb nxtmfp - offset.}
  mftype:      LongInt;                          {gdb mftype.}
  reccnt:      LongInt;                          {gdb reccnt.}
  mfcxx1:      LongInt;                          {gdb mfcxx1.}
  mfcxx2:      LongInt;                          {gdb mfcxx2 - MULTI: Data entry lock.}
  mfcxx3:      LongInt;                          {gdb mfcxx3 - MULTI: Exclusive write lock.}

End;


Type IsisRecDir = record

  tag:         LongInt;                          {field tag entry.}
  pos:         LongInt;                          {field position.}
  len:         LongInt;                          {field length entry.}

End;


Type IsisRecLeader = record

  mfn:         LongInt;                          {gdb mfn.}
  mfrl:        LongInt;                          {gdb mfrl - MULTI: record being updated.}
  mfbwb:       LongInt;                          {gdb mfbwb.}
  mfbwp:       LongInt;                          {gdb mfbwp - offset.}
  base:        LongInt;                          {gdb base (MSNVSPLT).}
  nvf:         LongInt;                          {gdb nvf.}
  status:LongInt;                          {gdb status.}

End;
```

2.3.2. Isis Space Structures.

```
Type IsisSpaHeader = record

  handle:      LongInt;                          {pointer to ISIS_SPACE.}
  name:        array [0..MAXPATHLEN] of Char;    {ISIS_SPACE name.}
  cipar:       array [0..MAXPATHLEN] of Char;    {cipar file name.}
  mf:          array [0..MAXPATHLEN] of Char;    {master file name.}
  ifi:         array [0..MAXPATHLEN] of Char;    {inverted file name.}
  fst:         array [0..MAXPATHLEN] of Char;    {fst file name.}
  pft:         array [0..MAXPATHLEN] of Char;    {pft file name.}
  fmt:         array [0..MAXPATHLEN] of Char;    {fmt file name.}
  stw:         array [0..MAXPATHLEN] of Char;    {stw file name.}
  fdt:         array [0..MAXPATHLEN] of Char;    {fdt file name.}
  isoin        array [0..MAXPATHLEN] of Char;    {import iso file name.}
  isoout       array [0..MAXPATHLEN] of Char;    {export iso file name.}
  giz:         array [0..MAXPATHLEN] of Char;    {gizmo file name.}
```

```
  dec:          array [0..MAXPATHLEN] of Char;      {decode file name.}
  rec:          LongInt;                            {number of RECSTRU shelves.}
  trm:          LongInt;                            {number of TRMSTRU shelves.}
  filestatus:   LongInt;                            {file status - bit mask.}

End;
```

2.3.3. Search Structures.

```
Type IsisSrcHeader = record

  number:       LongInt;                              {search number (start in 1).}
  recs:         LongInt                               {total records retrieved.}
  dbname:       array [0..MAXPATHLEN] of Char;        {data base name.}
  booleanexpr:  array [0.. SRC_EXPR_LENGTH] of Char; {search expression.}

End;


Type IsisSrcMfn = record

  mfn:          LongInt;                            {hit mfn component.}

End;


Type IsisSrcRegExp = record

  mfn:          LongInt;                              {hit mfn component.}
  tag:          LongInt;                              {hit tag component.}
 expr:          array [0.. SRC_EXPR_LENGTH] of Char; {regular expression.}

End;
```

2.3.4. Term Structures.

```
Type IsisTrmMfn = record

  mfn:          LongInt;                            {hit mfn component.}

End;


Type IsisTrmPosting = record

  posting:      LongInt;                            {current posting order.}
  mfn:          LongInt;                            {current posting pmfn.}
  tag:          LongInt;                            {current posting ptag.}
  occ:          LongInt;                            {current posting pocc.}
  cnt:          LongInt;                            {current posting pcnt.}
```

```
End;


Type IsisTrmRead = record

  key:              array [0.. KEY_LENGTH] of Char;         {term key.}

End;
```

2.4. Java Structures.

2.4.1. Record Structures.

```
public class IsisRecControl
{
   public int  ctlmfn                  //gdb ctlmfn.
   public int  nxtmfn;                 //gdb nxtmfn.
   public int  nxtmfb;                 //gdb nxtmfb.
   public int  nxtmfp;                 //gdb nxtmfp - offset.
   public int  mftype;                 //gdb mftypedef.
   public int  reccnt;                 //gdb reccnt.
   public int  mfcxx1;                 //gdb mfcxx1.
   public int  mfcxx2;                 //gdb mfcxx2 - MULTI: Data entry lock.
   public int  mfcxx3;                 //gdb mfcxx3 - MULTI: Exclusive write lock.
}

public class IsisRecDir
{
    public int  tag;                   //field tag entry.
    public int  pos;                   //field position.
    public int  len;                   //field length entry.
}

public class IsisRecLeader
{
   public int  mfn;                    //gdb mfn.
   public int  mfrl;                   //gdb mfrl - MULTI: record being updated.
   public int  mfbwb;                  //gdb mfbwb.
   public int  mfbwp;                  //gdb mfbwp - offset.
   public int  base;                   //gdb base (MSNVSPLT).
   public int  nvf;                    //gdb nvf.
   public int  status;                 //gdb status.
}
```

2.4.2. Isis Space Structures.

```
public class IsisSpaHeader
{
   public int      handle;          //pointer ISIS_SPACE.
   public String  name;             //ISIS_SPACE name.
   public String  cipar;            //cipar file name.
```

```
    public String  mf;              //master file name.
    public String  ifi;             //inverted file name.
    public String  fst;             //fst file name.
    public String  pft;             //pft file name.
    public String  fmt;             //fmt file name.
    public String  stw;             //stw file name.
    public String   fdt;            //fdt file name.
    public String  isoin;           //import iso file name.
    public String  isoout;          //export iso file name.
    public String  giz;             // gizmo file name.
    public String  dec;             // decode file name.
    public int      rec;            //number of RECSTRU shelves.
    public int      trm;            //number of TRMSTRU shelves.
    public int      filestatus;     //file status - bit mask.
}
```

2.4.3. Search Structures.

```
public class IsisSrcHeader
{
    public int      number;             //search number (start in 1).
    public int      recs;               //total records retrieved.
    public String  dbname;              //data base name.
    public String  booleanexpr;         //regular expression.
}

public class IsisSrcMfn
{
    public int  mfn;                    //hit mfn component.
}

public class IsisSrcRegExp
{
    public int  mfn;                    //hit mfn component.
    public int  tag;                    //hit tag component.
    public String  expr;                //search expression.
}
```

2.4.4. Term Structures.

```
public class IsisTrmMfn
{
    public int  mfn;                    //hit mfn component.
}

public class IsisTrmPosting
{
    public int  posting;                //current posting order.
    public int  mfn;                    //current posting pmfn.
    public int  tag;                    //current posting ptag.
    public int  occ;                    //current posting pocc.
    public int  cnt;                    //current posting pcnt.
}
```

```
public class IsisTrmRead
{
   public String key;              //key term.
}
```

3. ISIS_DLL Error Codes.

```
const ZERO                  =  0;
```

3.1. Data Base Errors.

```
const ERR_DBDELOCK          =  -101;        /* Data Base access denied (data entry lock) */
const ERR_DBEWLOCK          =  -102;        /* Data Base access denied (probably exclusive
                                               write lock). */
const ERR_DBMONOUSR         =  -103;        /* Data Base access is single-user. */
const ERR_DBMULTUSR         =  -104;        /* Data Base access is multi-user. */
```

3.2. File Manipulation Errors.

```
const ERR_FILECREATE        =  -201;        /* File create error. */
const ERR_FILEDELETE        =  -202;        /* File delete error. */
const ERR_FILEEMPTY         =  -203;        /* File (empty). */
const ERR_FILEFLUSH         =  -204;        /* File flush error. */
const ERR_FILEFMT           =  -205;        /* File does not exist (fmt). */
const ERR_FILEFST           =  -206;        /* File does not exist (fst). */
const ERR_FILEINVERT        =  -207;        /* File does not exist (inverted). */
const ERR_FILEISO           =  -208;        /* File does not exist (ISO). */
const ERR_FILEMASTER        =  -209;        /* File does not exist (master). */
const ERR_FILEMISSING       =  -210;        /* File missing. */
const ERR_FILEOPEN          =  -211;        /* File open error. */
const ERR_FILEPFT           =  -212;        /* File does not exist (pft). */
const ERR_FILEREAD          =  -213;        /* File read error. */
const ERR_FILERENAME        =  -214;        /* File rename error. */
const ERR_FILESTW           =  -215;        /* File does not exist (stw). */
const ERR_FILEWRITE         =  -216;        /* File write error. */
const ERR_FILEEOF           =  -217;    /* File end. */
```

3.3. Low Level Engine Errors.

```
const ERR_LLCISISETRAP      =  -301;        /* Cisis Low Level Error Trap. */
const ERR_LLISISETRAP       =  -302;        /* Isis Low Level Error Trap. */
const ERR_LLJISISETRAP      =  -303;        /* Jisis Low Level Error Trap. */
const ERR_LLCORBAETRAP      =  -304;        /* Corba Low Level Error Trap. */
```

3.4. Memory Manipulation Errors.

```
const ERR_MEMALLOCAT     = -401;          /* Memory Allocation Error. */
```

3.5. Parameter Specification Errors.

```
const ERR_PARAPPHAND     = -501;          /* Invalid application handle. */
const ERR_PARFILNSIZ     = -502;          /* Invalid file name size. */
const ERR_PARFLDSYNT     = -503;          /* Syntax Error (field update). */
const ERR_PARFMTSYNT     = -504;          /* Syntax Error (format). */
const ERR_PARNULLPNT     = -505;          /* NULL pointer. */
const ERR_PARNULLSTR     = -506;          /* String with zero size. */
const ERR_PAROUTRANG     = -507;          /* Parameter out of range. */
const ERR_PARSPAHAND     = -508;          /* Invalid space handle.*/
const ERR_PARSRCSYNT     = -509;          /* Syntax Error (search). */
const ERR_PARSUBFSPC     = -510;          /* Invalid subfield specification. */
const ERR_PARUPDSYNT     = -511;          /* Syntax Error (record update). */
```

3.6. Record Errors.

```
const ERR_RECEOF         = -601;          /* Record eof: found eof in data base. */
const ERR_RECLOCKED      = -602;          /* Record locked. */
const ERR_RECLOGIDEL     = -603;          /* Record logically deleted. */
const ERR_RECNOTNORM     = -604;          /* Record condition is not RCNORMAL. */
const ERR_RECPHYSDEL     = -605;          /* Record physically deleted. */
```

3.7. Term Errors.

```
const ERR_TRMEOF         = -701;          /* Term eof: found eof in data base. */
const ERR_TRMNEXT        = -702;          /* Term next: key not found. */
```

3.8. Unexpected Errors.

```
const ERR_UNEXPECTED     = -999;          /* Unexpected Error. */
```

Annex 2 - ISIS_DLL functions prototypes

1. C functions prototypes.

1.1. Application functions.

```
long
CALLBACK
        IsisAppAcTab (long  apphandle,
                      char *actab);

long
CALLBACK
        IsisAppDebug (long apphandle,
                      long debugflag);

long
CALLBACK
        IsisAppDelete (long apphandle);

long
CALLBACK
        IsisAppLogFile (long   apphandle,
                        char   *filename);

long
CALLBACK
        IsisAppNew ( );

long
CALLBACK
        IsisAppParGet (long  apphandle,
                       char *parpinp,
                       char *parpoutp,
                       long   areasize);

long
CALLBACK
        IsisAppParSet (long  apphandle,
                       char *appareap);

long
CALLBACK
        IsisAppUcTab (long  apphandle,
                      char *uctab);
```

1.2. DLL functions.

```
float
CALLBACK
        IsisDllVersion ( );
```

1.3. Link functions.

```
long
CALLBACK
        IsisLnkIfLoad (long handle);

long
CALLBACK
        IsisLnkIfLoadEx (long handle,
                         long  reset,
                         long  posts,
                         long  balan);

long
CALLBACK
        IsisLnkSort (long handle);
```

1.4. Record functions.

```
long
CALLBACK
        IsisRecControlMap (long   handle,
                           char *ctrl);

long
CALLBACK
        IsisRecCopy (long  handle_from,
                     long  index_from,
                     long  handle_to,
                     long  index_to);

long
CALLBACK
        IsisRecDirMap (long   handle,
                       long   index,
                       long   firstpos,
                       long   lastpos,
                       char  *dir);

long
CALLBACK
        IsisRecDummy (long  handle,
                      long  index);

long
CALLBACK
        IsisRecDump (long   handle,
                     long   index,
                     char  *dump,
                      long   areasize);

long
CALLBACK
        IsisRecField (long   handle,
                      long   index,
                      long   tag,
                      long   occ,
                      char  *field_area,
                      long   areasize);
```

```
long
CALLBACK
        IsisRecFieldN (long  handle,
                       long  index,
                       long  pos,
                       char  *field_area,
                       long  areasize);

long
CALLBACK
        IsisRecFieldOcc (long  handle,
                         long  index,
                         long  tag);

long
CALLBACK
        IsisRecFieldUpdate (long  handle,
                            long  index,
                            char *fldupd);

long
CALLBACK
        IsisRecFormat (long  handle,
                       long   index,
                       char  *areap,
                       long   areasize);


long
CALLBACK
        IsisRecFormatEx (long  handle,
                         long  index,
                         long  linesize,
                         char  *areap,
                         long  areasize);

long
CALLBACK
        IsisRecIfUpdate (long handle,
                         long mfn);

long
CALLBACK
        IsisRecIfUpdateEx (long handle,
                           long beginmfn,
                           long endmfn,
                           long  keeppending);

long
CALLBACK
        IsisRecIsoRead (long handle,
                        long  index);

long
CALLBACK
        IsisRecIsoWrite (long handle,
                         long  index);
```

```
long
CALLBACK
        IsisRecLeaderMap (long  handle,
                          long  index,
                          char *leader);

long
CALLBACK
        IsisRecLnk (long handle,
                    long beginmfn,
                    long endmfn);

long
CALLBACK
        IsisRecLockRecall (long  handle,
                           long  index,
                           long  mfn,
                           long  tag,
                           char* password);

long
CALLBACK
        IsisRecMerge (long   handle_from,
                      long  index_from,
                      long  handle_to,
                      long  index_to);

long
CALLBACK
        IsisRecMfn (long handle,
                    long  index);

long
CALLBACK
        IsisRecMfnChange (long  handle,
                          long   index,
                          long  mfn);

long
CALLBACK
        IsisRecNew (long  handle,
                    long  index);

long
CALLBACK
        IsisRecNewLock (long  handle,
                        long  index);

long
CALLBACK
        IsisRecNvf (long handle,
                    long  index);

long
CALLBACK
        IsisRecRead (long  handle,
                     long  index,
```

```
                        long  mfn);

long
CALLBACK
        IsisRecReadLock (long handle,
                         long  index,
                         long  mfn);

long
CALLBACK
        IsisRecShelfSize (long  handle,
                          long  index,
                          long  mem);

long
CALLBACK
        IsisRecSubField (long   handle,
                         long    index,
                         long    tag,
                         long    fldocc,
                         char  *subfield,
                         char  *subfield_area,
                         long    areasize);

long
CALLBACK
        IsisRecSubFieldEx (long  handle,
                           long   index,
                           long    tag,
                           long    fldocc,
                           char  *subfield,
                           long    subfldocc,
                           char  *subfield_area,
                           long    areasize);

long
CALLBACK
        IsisRecUndelete (long  handle,
                         long  index);

long
CALLBACK
        IsisRecUnlock (long  handle,
                       long  index);

long
CALLBACK
        IsisRecUnlockForce (long handle,
                            long index);

long
CALLBACK
         IsisRecUpdate (long  handle,
                        long   index,
                        char  *sparser);

long
CALLBACK
```

```
        IsisRecWrite (long  handle,
                      long  index);

long
CALLBACK
        IsisRecWriteLock (long  handle,
                          long index);

long
CALLBACK
        IsisRecWriteUnlock (long  handle,
                            long  index);
```

1.5. Space functions.

```
long
CALLBACK
        IsisSpaDb (long  handle,
                   char *dbname);

long
CALLBACK
        IsisSpaDelete (long  handle);

long
CALLBACK
        IsisSpaFdt (long   handle,
                    char  *fdtname);


long
CALLBACK
        IsisSpaFst (long  handle,
                    char *fstname);

ErrorCode
CALLBACK
       IsisSpaGf (long  handle,
                  char *gizname);

long
CALLBACK
        IsisSpaHeaderMap (long   handle,
                          char  *harea);

long
CALLBACK
        IsisSpaIf (long   handle,
                   char  *iname);

long
CALLBACK
        IsisSpaIfCreate (long  handle);

long
CALLBACK
        IsisSpaIsoDelim (long  handle,
```

```
                                char *recdelim,
                                char *fielddelim);

long
CALLBACK
        IsisSpaIsoIn (long  handle,
                      char *filename);

long
CALLBACK
        IsisSpaIsoOut (long  handle,
                       char *filename);

long
CALLBACK
        IsisSpaIsoOutCreate (long handle);

long
CALLBACK
        IsisSpaMf (long   handle,
                   char  *mname);

long
CALLBACK
        IsisSpaMfCreate (long  handle);

long
CALLBACK
        IsisSpaMfUnlockForce (long handle);

long
CALLBACK
        IsisSpaName (long  handle,
                     char *sname);

long
CALLBACK
        IsisSpaNew (long  apphandle);

long
CALLBACK
        IsisSpaPft (long  handle,
                    char *format);

long
CALLBACK
        IsisSpaRecDelim (long   handle,
                         char  *begindelim,
                         char  *enddelim);

long
CALLBACK
        IsisSpaRecShelves (long  handle,
                           long  max_mst);

long
CALLBACK
        IsisSpaStw (long   handle,
```

```
                        char *stwname);

long
CALLBACK
        IsisSpaTrmShelves (long handle,
                           long  max_trm);
```

1.6. Search functions.

```
long
CALLBACK
        IsisSrcHeaderMap (long  apphandle,
                          long   tsfnum,
                          long   searchnum,
                          char  *sstrup);
long
CALLBACK
        IsisSrcLogFileFlush (long  apphandle,
                             long   tsfnum);
long
CALLBACK
        IsisSrcLogFileSave (long  apphandle,
                            long    tsfnum,
                            char  *filename);
long
CALLBACK
        IsisSrcLogFileUse (long  apphandle,
                           long   tsfnum,
                           char  *filename);

long
CALLBACK
        IsisSrcMfnMap (long  apphandle,
                       long    tsfnum,
                       long    searchnum,
                       long    firstpos,
                       long    lastpos,
                       char  *mfnareap);

long
CALLBACK
        IsisSrcRegExpMap(long  handle,
                         char *expr,
                         long  mfnb,
                         long  mfne,
                         char *areap,
                         long  areasize);

long
CALLBACK
        IsisSrcSearch (long   handle,
                       long   tsfnum,
                       char *express,
                       char *areap);

long
CALLBACK
```

```
IsisSrcSearchEx(long  handle,
                long  tsfnum,
                char *itable,
                char *express,
                char *areap);
```

1.7. Term functions.

```
long
CALLBACK
        IsisTrmMfnMap (long  handle,
                       long   index_trm,
                       long   firstpos,
                       long   lastpos,
                       char *mfnareap);

long
CALLBACK
        IsisTrmPostingMap (long  handle,
                           long   index_trm,
                           long   firstpos,
                           long   lastpos,
                           char *parea);

long
CALLBACK
        IsisTrmReadMap (long  handle,
                        long   index_trm,
                        char *key);

long
CALLBACK
        IsisTrmReadNext (long  handle,
                         long  index_trm,
                         char *key);

long
CALLBACK
        IsisTrmReadPrevious (long  handle,
                             long  index_trm,
                             char *prefix,
                             char *key);

long
CALLBACK
        IsisTrmShelfSize (long  handle,
                          long   index_trm,
                          long   mem);
```

2. Visual Basic functions prototypes.

2.1. Application functions.

```
Declare Function IsisAppAcTab Lib "isis32.dll" (ByVal AppHandle&, ByVal AcTab$) As Long
Declare Function IsisAppDebug Lib "isis32.dll" (ByVal AppHandle&, ByVal Flag&) As Long
```

```
Declare Function IsisAppDelete Lib "isis32.dll" (ByVal AppHandle&) As Long
Declare Function IsisAppLogFile Lib "isis32.dll" (ByVal AppHandle&, ByVal FileName$) As Long
Declare Function IsisAppNew Lib "isis32.dll" () As Long
Declare Function IsisAppParGet Lib "isis32.dll" (ByVal AppHandle&, ByVal ParIn$, ByVal
ParOut$, ByVal AreaSize&) As Long
Declare Function IsisAppParSet Lib "isis32.dll" (ByVal AppHandle&, ByVal AppAreap$) As Long
Declare Function IsisAppUcTab Lib "isis32.dll" (ByVal AppHandle&, ByVal UcTab$) As Long
```

2.2. DLL functions.

```
Declare Function IsisDllVersion Lib "isis32.dll" () As Single
```

2.3. Link functions.

```
Declare Function IsisLnkIfLoad Lib "isis32.dll" (ByVal Handle&) As Long

Declare Function IsisLnkIfLoadEx Lib "isis32.dll" (ByVal Handle&, ByVal ResetFlag&, ByVal
Posts&, ByVal Balan&) As Long

Declare Function IsisLnkSort Lib "isis32.dll" (ByVal Handle&) As Long
```

2.4. Record functions.

```
Declare Function IsisRecControlMap Lib "isis32.dll" (ByVal Handle&, P As IsisRecControl) As
Long

Declare Function IsisRecCopy Lib "isis32.dll" (ByVal HandleFrom&, ByVal IndexFrom&, ByVal
HandleTo&, ByVal IndexTo&) As Long

Declare Function IsisRecDirMap Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal FirstPos&,
ByVal LastPos&, P As IsisRecDir) As Long

Declare Function IsisRecDummy Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecDump Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal
DumpArea$, ByVal AreaSize&) As Long

Declare Function IsisRecField Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Tag&, ByVal
Occ&, ByVal FieldArea$, ByVal AreaSize&) As Long

Declare Function IsisRecFieldN Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Pos&,
ByVal FieldArea$, ByVal AreaSize&) As Long

Declare Function IsisRecFieldOcc Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Tag&) As
Long

Declare Function IsisRecFieldUpdate Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal
FldUpd$) As Long

Declare Function IsisRecFormat Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Farea$,
ByVal AreaSize&) As Long

Declare Function IsisRecFormatEx Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal
LineSize&, ByVal Farea$, ByVal AreaSize&) As Long
```

```
Declare Function IsisRecIfUpdate Lib "isis32.dll" (ByVal Handle&, ByVal Mfn&) As Long

Declare Function IsisRecIfUpdateEx Lib "isis32.dll" (ByVal Handle&, ByVal BeginMfn&, ByVal
EndMfn&, ByVal KeepPending&) As Long

Declare Function IsisRecIsoRead Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecIsoWrite Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecLeaderMap Lib "isis32.dll" (ByVal Handle&, ByVal Index&, P As
IsisRecLeader) As Long

Declare Function IsisRecLnk Lib "isis32.dll" (ByVal Handle&, ByVal BeginMfn&, ByVal EndMfn&)
As Long

Declare Function IsisRecLockRecall Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Mfn&,
ByVal Tag&, ByVal Password$) As Long

Declare Function IsisRecMerge Lib "isis32.dll" (ByVal HandleFrom&, ByVal IndexFrom&, ByVal
HandleTo&, ByVal IndexTo&) As Long

Declare Function IsisRecMfn Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecMfnChange Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Mfn&)
As Long

Declare Function IsisRecNew Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecNewLock Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecNvf Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecRead Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Mfn&) As
Long

Declare Function IsisRecReadLock Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Mfn&)
As Long

Declare Function IsisRecShelfSize Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal
Memory&) As Long

Declare Function IsisRecSubField Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Tag&,
ByVal FldOcc&, ByVal SubField$, ByVal SubFieldArea$, ByVal AreaSize&) As Long

Declare Function IsisRecSubFieldEx Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Tag&,
ByVal FldOcc&, ByVal SubField$, ByVal SubFldOcc&, ByVal SubFieldArea$, ByVal AreaSize&)
As Long

Declare Function IsisRecUndelete Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecUnlock Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecUnlockForce Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecUpdate Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal
SpecArea$) As Long
```

```
Declare Function IsisRecWrite Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecWriteLock Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long

Declare Function IsisRecWriteUnlock Lib "isis32.dll" (ByVal Handle&, ByVal Index&) As Long
```

2.5. Space functions.

```
Declare Function IsisSpaDb Lib "isis32.dll" (ByVal Handle&, ByVal NameDb$) As Long

Declare Function IsisSpaDelete Lib "isis32.dll" (ByVal Handle&) As Long

Declare Function IsisSpaDf Lib "isis32.dll" (ByVal Handle&, ByVal NameDf$) As Long

Declare Function IsisSpaFdt Lib "isis32.dll" (ByVal Handle&, ByVal NameFdt$) As Long

Declare Function IsisSpaFst Lib "isis32.dll" (ByVal Handle&, ByVal NameFst$) As Long

Declare Function IsisSpaGf Lib "isis32.dll" (ByVal Handle&, ByVal NameGf$) As Long

Declare Function IsisSpaHeaderMap Lib "isis32.dll" (ByVal Handle&, P As IsisSpaHeader) As
Long

Declare Function IsisSpaIf Lib "isis32.dll" (ByVal Handle&, ByVal NameIf$) As Long

Declare Function IsisSpaIfCreate Lib "isis32.dll" (ByVal Handle&) As Long

Declare Function IsisSpaIsoDelim Lib "isis32.dll" (ByVal Handle&, ByVal RecDelim$, ByVal
FieldDelim$) As Long

Declare Function IsisSpaIsoIn Lib "isis32.dll" (ByVal Handle&, ByVal FileName$) As Long

Declare Function IsisSpaIsoOut Lib "isis32.dll" (ByVal Handle&, ByVal FileName$) As Long

Declare Function IsisSpaIsoOutCreate Lib "isis32.dll" (ByVal Handle&) As Long

Declare Function IsisSpaMf Lib "isis32.dll" (ByVal Handle&, ByVal NameMst$) As Long

Declare Function IsisSpaMfCreate Lib "isis32.dll" (ByVal Handle&) As Long

Declare Function IsisSpaMfUnlockForce Lib "isis32.dll" (ByVal Handle&) As Long

Declare Function IsisSpaName Lib "isis32.dll" (ByVal Handle&, ByVal NameSpace$) As Long

Declare Function IsisSpaNew Lib "isis32.dll" (ByVal AppHandle&) As Long

Declare Function IsisSpaPft Lib "isis32.dll" (ByVal Handle&, ByVal NamePft$) As Long

Declare Function IsisSpaRecDelim Lib "isis32.dll" (ByVal Handle&, ByVal BeginDelim$, ByVal
EndDelim$) As Long

Declare Function IsisSpaRecShelves Lib "isis32.dll" (ByVal Handle&, ByVal MaxMst&) As Long

Declare Function IsisSpaStw Lib "isis32.dll" (ByVal Handle&, ByVal NameStw$) As Long

Declare Function IsisSpaTrmShelves Lib "isis32.dll" (ByVal Handle&, ByVal MaxMst&) As Long
```

2.6. Search functions.

```
Declare Function IsisSrcHeaderMap Lib "isis32.dll" (ByVal AppHandle&, ByVal TSFNum&, ByVal SearchNo&, P As IsisSrcHeader) As Long

Declare Function IsisSrcLogFileFlush Lib "isis32.dll" (ByVal AppHandle&, ByVal TSFNum&) As Long

Declare Function IsisSrcLogFileSave Lib "isis32.dll" (ByVal AppHandle&, ByVal TSFNum&, ByVal FileName$) As Long

Declare Function IsisSrcLogFileUse Lib "isis32.dll" (ByVal AppHandle&, ByVal TSFNum&, ByVal FileName$) As Long

Declare Function IsisSrcMfnMap Lib "isis32.dll" (ByVal AppHandle&, ByVal TSFNum&, ByVal SearchNo&, ByVal FirstPos&, ByVal LastPos&, P As IsisSrcMfn) As Long

Declare Function IsisSrcRegExpMap Lib "isis32.dll" (ByVal Handle&, ByVal Expr$, ByVal Mfmb&, ByVal MfnE&, P As IsisSrcRegExp, ByVal Size&) As Long

Declare Function IsisSrcSearch Lib "isis32.dll" (ByVal Handle&, ByVal TSFNum&, ByVal Bool$, P As IsisSrcHeader) As Long

Declare Function IsisSrcSearchEx Lib "isis32.dll" (ByVal Handle&, ByVal TSFNum&, ByVal ITable$, ByVal Bool$, P As IsisSrcHeader) As Long
```

2.7. Term functions.

```
Declare Function IsisTrmMfnMap Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal FirstPos&, ByVal LastPos&, P As IsisTrmMfn) As Long

Declare Function IsisTrmPostingMap Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal FirstPos&, ByVal LastPos&, P As IsisTrmPosting) As Long

Declare Function IsisTrmReadMap Lib "isis32.dll" (ByVal Handle&, ByVal Index&, P As IsisTrmRead) As Long

Declare Function IsisTrmReadNext Lib "isis32.dll" (ByVal Handle&, ByVal Index&, P As IsisTrmRead) As Long

Declare Function IsisTrmReadPrevious Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Prefix$, P As IsisTrmRead) As Long

Declare Function IsisTrmShelfSize Lib "isis32.dll" (ByVal Handle&, ByVal Index&, ByVal Memory&) As Long
```

2.8. General functions.

```
Declare Function OemToCharBuff Lib "user32" Alias "OemToCharBuffA" (ByVal lpszSrc As String, ByVal lpszDst As String, ByVal cchDstLength As Long) As Long

Declare Function CharToOemBuff Lib "user32" Alias "CharToOemBuffA" (ByVal lpszSrc As String, ByVal lpszDst As String, ByVal cchDstLength As Long) As Long
```

```
Declare Function SetHandleCount Lib "kernel32" (ByVal wNumber As Long) As Long
```

3. Delphi functions prototypes.

3.1. Application functions.

```
Function IsisAppAcTab(AppHandle:LongInt;AcTab:PChar):LongInt;          external 'isis32.dll';

Function IsisAppDebug(AppHandle:LongInt;Flag:LongInt):LongInt;          external 'isis32.dll';
Function IsisAppDelete(AppHandle:LongInt):LongInt;       external 'isis32.dll';
Function IsisAppLogFile(AppHandle:LongInt;FileName:PChar):LongInt;    external 'isis32.dll';

Function IsisAppNew:LongInt;     external 'isis32.dll';
FunctionIsisAppParGet(AppHandle:LongInt;ParIn:PChar;ParOut:PChar;AreaSize:LongInt):
LongInt;              external 'isis32.dll';
Function IsisAppParSet(AppHandle:LongInt;AppArea:PChar):LongInt;
            external 'isis32.dll';

Function IsisAppUcTab(AppHandle:LongInt;UcTab:PChar):LongInt;           external 'isis32.dll';
```

3.2. DLL functions.

```
Function IsisDllVersion:single;
            external ‘isis32’;
```

3.3. Link functions.

```
Function IsisLnkIfLoad(Handle:LongInt):LongInt;
            external 'isis32.dll';
 Function IsisLnkIfLoadEx(Handle:LongInt;Reset:LongInt;Posts:LongInt;Balan:LongInt):LongInt;
external 'isis32.dll';
 Function IsisLnkSort(Handle:LongInt):LongInt;    external 'isis32.dll';
```

3.4. Record functions.

```
Function IsisRecControlMap(Handle:LongInt;var P:IsisRecControl):LongInt;          external
'isis32.dll';
Function IsisRecCopy(HandleFrom:LongInt;IndexFrom:LongInt;HandleTo:LongInt;
IndexTo:LongInt ):LongInt;          external 'isis32.dll';
Function IsisRecDirMap(Handle:LongInt;Index:LongInt;FirstPos:LongInt;LastPos:LongInt;var
P:IsisRecDir):LongInt;
            external 'isis32.dll';
Function IsisRecDummy(Handle:LongInt;Index:LongInt):LongInt;              external 'isis32.dll';
Function IsisRecDump(Handle:LongInt;Index:LongInt;FieldArea:PChar;AreaSize:LongInt):
LongInt;
            external 'isis32.dll';
Function IsisRecField(Handle:LongInt;Index:LongInt;Tag:LongInt;Occ:LongInt;FieldArea:PChar;
AreaSize: LongInt):LongInt;         external 'isis32.dll';
Function IsisRecFieldN(Handle:LongInt;Index:LongInt;Pos:LongInt;FieldArea:PChar;
AreaSize:LongInt):LongInt;
            external 'isis32.dll';
```

```
Function IsisRecFieldOcc(Handle:LongInt;Index:LongInt;Tag:LongInt):LongInt;     external
'isis32.dll';
Function IsisRecFieldUpdate(Handle:LongInt;Index:LongInt;FldUpd:PChar):LongInt;
        external 'isis32.dll';

Function IsisRecFormat(Handle:LongInt;Index:LongInt;Farea:PChar;AreaSize:LongInt):LongInt;
external 'isis32.dll';
Function IsisRecFormatEx(Handle:LongInt;Index:LongInt;LineSize:LongInt;Farea:PChar;
AreaSize:LongInt):LongInt;
        external 'isis32.dll';

Function IsisRecIfUpdate(Handle:LongInt;Mfn:LongInt):LongInt;   external 'isis32.dll';
Function IsisRecIfUpdateEx(Handle:LongInt;BeginMfn:LongInt;EndMfn:LongInt;
KeepPending:LongInt):LongInt;
        external 'isis32.dll';

Function IsisRecIsoRead(Handle:LongInt;Index:LongInt):LongInt;          external 'isis32.dll';
Function IsisRecIsoWrite(Handle:LongInt; Index:LongInt):LongInt;        external 'isis32.dll';
Function IsisRecLeaderMap(Handle:LongInt;Index:LongInt;var P:IsisRecLeader):LongInt;
        external 'isis32.dll';

Function IsisRecLnk(Handle:LongInt;BeginMfn:LongInt;EndMfn:LongInt):LongInt;            external
'isis32.dll';
Function IsisRecLockRecall(Handle:LongInt;Index:LongInt;Mfn:LongInt;Tag:LongInt;
Password:PChar):LongInt;
        external 'isis32.dll';

Function IsisRecMerge(HandleFrom:LongInt;IndexFrom:LongInt;HandleTo:LongInt;
IndexTo:LongInt):LongInt;       external 'isis32.dll';
Function IsisRecMfn(Handle:LongInt;Index:LongInt):LongInt;
        external 'isis32.dll';

Function IsisRecMfnChange(Handle:LongInt;Index:LongInt;Mfn:LongInt):LongInt;            external
'isis32.dll';
Function IsisRecNew(Handle:LongInt;Index:LongInt):LongInt;      external 'isis32.dll';
Function IsisRecNewLock(Handle:LongInt;Index:LongInt):LongInt;
        external 'isis32.dll';

Function IsisRecNvf(Handle:LongInt;Index:LongInt):LongInt;      external 'isis32.dll';
Function IsisRecRead(Handle:LongInt;Index:LongInt;Mfn:LongInt):LongInt;         external
'isis32.dll';

Function IsisRecReadLock(Handle:LongInt;Index:LongInt;Mfn:LongInt):LongInt;   external
'isis32.dll';
Function IsisRecShelfSize(Handle:LongInt;Index:LongInt;Memory:LongInt):LongInt;         external
'isis32.dll';
Function IsisRecSubField(Handle:LongInt;Index:LongInt;Tag:LongInt;FldOcc:LongInt;
Subfield:PChar;SubFieldArea:PChar;AreaSize:LongInt):LongInt;
        external 'isis32.dll';

Function IsisRecSubFieldEx(Handle:LongInt;Index:LongInt;Tag:LongInt;FldOcc:LongInt;
Subfield:PChar;SubFldOcc:LongInt;SubFieldArea:PChar;AreaSize:LongInt):LongInt;
        external 'isis32.dll';
 Function IsisRecUndelete(Handle:LongInt;Index:LongInt):LongInt;         external 'isis32.dll';
 Function IsisRecUnlock(Handle:LongInt;Index:LongInt):LongInt;           external 'isis32.dll';

 Function IsisRecUnlockForce(Handle:LongInt;Index:LongInt):LongInt;    external 'isis32.dll';
```

```
Function IsisRecUpdate(Handle:LongInt;Index:LongInt;FieldArea:PChar):LongInt;          external
'isis32.dll';
Function IsisRecWrite(Handle:LongInt;Index:LongInt):LongInt;
        external 'isis32.dll';

Function IsisRecWriteLock(Handle:LongInt;Index:LongInt):LongInt;       external 'isis32.dll';
Function IsisRecWriteUnlock(Handle:LongInt;Index:LongInt):LongInt;     external 'isis32.dll';
```

3.5. Space functions.

```
Function IsisSpaDb(Handle:LongInt;NameBD:PChar):LongInt;    external 'isis32.dll';
Function IsisSpaDelete(Handle:LongInt):LongInt;          external 'isis32.dll';
Function IsisSpaFdt(Handle:LongInt;NameFdt:PChar):LongInt;   external 'isis32.dll';
Function IsisSpaFst(Handle:LongInt;NameFst:PChar):LongInt;
        external 'isis32.dll';
      Function IsisSpaGf(Handle:LongInt;NameGf:PChar):LongInt;     external 'isis32.dll';
Function IsisSpaHeaderMap(Handle:LongInt;var P:IsisSpaHeader):LongInt;       external
'isis32.dll';
Function IsisSpaIf(Handle:LongInt;NameIf:PChar):LongInt;         external 'isis32.dll';
Function IsisSpaIfCreate(Handle:LongInt):LongInt;        external 'isis32.dll';
Function IsisSpaIsoDelim(Handle:LongInt;RecDelim:PChar;FieldDelim:PChar):LongInt;
        external 'isis32.dll';
Function IsisSpaIsoIn(Handle:LongInt;FileName:PChar):LongInt;          external 'isis32.dll';
Function IsisSpaIsoOut(Handle:LongInt;FileName:PChar):LongInt;         external 'isis32.dll';
Function IsisSpaIsoOutCreate(Handle:LongInt):LongInt;
        external 'isis32.dll';
Function IsisSpaMf(Handle:LongInt;NameMst:PChar):LongInt;    external 'isis32.dll';

Function IsisSpaMfCreate(Handle:LongInt):LongInt;        external 'isis32.dll';
Function IsisSpaMfUnlockForce(Handle:LongInt):LongInt;
        external 'isis32.dll';
Function IsisSpaName(Handle:LongInt;NameSpace:PChar):LongInt;      external 'isis32.dll';
Function IsisSpaNew(AppHandle:LongInt):LongInt;          external 'isis32.dll';
Function IsisSpaPft(Handle:LongInt;NamePft:PChar):LongInt;
        external 'isis32.dll';
Function IsisSpaRecDelim(Handle:LongInt;BeginDelim:PChar;EndDelim:PChar):LongInt;
external 'isis32.dll';
Function IsisSpaRecShelves(Handle:LongInt;MaxMst:LongInt):LongInt;   external 'isis32.dll';
Function IsisSpaStw(Handle:LongInt;NameStw:PChar):LongInt;
        external 'isis32.dll';
Function IsisSpaTrmShelves(Handle:LongInt;MaxMst:LongInt):LongInt;   external 'isis32.dll';
```

3.6. Search functions.

```
Function IsisSrcHeaderMap(AppHandle:LongInt;TSFNum:LongInt;SearchNo:LongInt;var
P:IsisSrcHeader):LongInt;
        external 'isis32.dll';
Function IsisSrcLogFileFlush(AppHandle:LongInt;TSFNum:LongInt):LongInt;      external
'isis32.dll';
Function IsisSrcLogFileSave(AppHandle:LongInt;TSFNum:LongInt;FileName:PChar):LongInt;
        external 'isis32.dll';
Function IsisSrcLogFileUse(AppHandle:LongInt;TSFNum:LongInt;FileName:PChar):LongInt;
external 'isis32.dll';
```

```
Function sisSrcMfnMap(AppHandle:LongInt;TSFNum:LongInt;SearchNo:LongInt;
FirstPos:LongInt;LastPos:LongInt;var P:IsisSrcMfn):LongInt;
        external 'isis32.dll';
Function IsisSrcRegExpMap(Handle:LongInt;Boolean:PChar;MfnB:LongInt;MfnE:LongInt;var
P:IsisSrcRegExp;Size:LongInt):LongInt;  external 'isis32.dll';
Function IsisSrcSearch(Handle:LongInt;TSFNum:LongInt;Boolean:PChar;var
P:IsisSrcHeader):LongInt;
        external 'isis32.dll';

Function IsisSrcSearchEx(Handle:LongInt;TSFNum:LongInt;Table:PChar;Boolean:PChar;var
P:IsisSrcHeader):LongInt;           external 'isis32.dll';
```

3.7. Term functions.

```
Function IsisTrmMfnMap(Handle:LongInt;Index:LongInt;FirstPos:LongInt;LastPos:LongInt;var
P:IsisTrmMfn):LongInt;
        external 'isis32.dll';
Function IsisTrmPostingMap(Handle:LongInt;Index:LongInt;FirstPos:LongInt;LastPos:LongInt;
                var P:IsisTrmPosting):LongInt;
        external 'isis32.dll';
Function IsisTrmReadMap(Handle:LongInt;Index:LongInt;var P:IsisTrmRead):LongInt;    external
'isis32.dll';
Function IsisTrmReadNext(Handle:LongInt;Index:LongInt;var P:IsisTrmRead):LongInt;    external
'isis32.dll';
Function IsisTrmReadPrevious(Handle:LongInt;Index:LongInt;Prefix:Pchar; var P:IsisTrmRead):
LongInt;
        external 'isis32.dll';
Function IsisTrmShelfSize(Handle:LongInt;Index:LongInt;Memory:LongInt):LongInt;        external
'isis32.dll';
```

3.8. General functions.

```
Function OemToAnsiBuff(lpszSrc:PChar;lpszDst:PChar;cchDstLength:Integer):LongInt;
        external 'Keyboard';

Function AnsiToOemBuff(lpszSrc:PChar;lpszDst:PChar;cchDstLength:Integer):LongInt;
        external 'Keyboard';

Function SetHandleCount(wNumber:Integer):LongInt;
        external 'Kernel';
```

3. Java functions prototypes.

```
public class ISISAPI implements Constants, ErrorCodes{
 // 3.1. Application functions.

 public static native int IsisAppAcTab (int    handle,                                    String
actab);

 public static native int IsisAppDebug (int apphandle,
                                        int debugflag);

 public static native int IsisAppDelete (int apphandle);
```

```
public static native int IsisAppLogFile (int    apphandle,
                                         String filename);

public static native int IsisAppNew ( );

public static native int IsisAppParGet (int    apphandle,
                                        String ipar,
                                        String opar[],
                                        int    areasize);

public static native int IsisAppParSet (int    apphandle,
                                        String apparea);

public static native int IsisAppUcTab (int    apphandle,
                                       String uctab);


// 3.2. DLL functions.
public static native float IsisDllVersion ( );


// 3.3. Link functions.


public static native int IsisLnkIfLoad (int handle); public static native int IsisLnkIfLoadEx (int
handle,                                          int reset,
                                          int posts,
                                          int balan);

public static native int IsisLnkSort (int handle);


// 3.4. Record functions.


public static native int IsisRecControlMap (int           handle,
                                            IsisRecControl ctrl);

public static native int IsisRecCopy (int handle_from,
                                      int index_from,
                                      int handle_to,
                                      int index_to);

public static native int IsisRecDirMap (int       handle,
                                        int       index,
                                        int       firstpos,
                                        int       lastpos,
                                        IsisRecDir dir[]);

public static native int IsisRecDummy (int handle,
                                       int index);

public static native int IsisRecDump (int    handle,
                                      int    index,
                                      String dump[],
                                      int    areasize);

public static native int IsisRecField (int    handle,
```

```
                                        int    index,
                                        int    tag,
                                        int    occ,
                                        String field_area[],
                                        int    areasize);

public static native int IsisRecFieldN (int    handle,
                                        int    index,
                                        int    pos,
                                        String field_area[],
                                        int    areasize);

public static native int IsisRecFieldOcc (int handle,
                                          int index,
                                          int tag);

public static native int IsisRecFieldUpdate (int    handle,
                                             int    index,
                                             String fldupd);

public static native int IsisRecFormat (int    handle,
                                        int    index,
                                        String farea[],
                                        int    areasize);

public static native int IsisRecFormatEx (int    handle,
                                          int    index,
                                          int    linesize,
                                          String farea[],
                                          int    areasize);

public static native int IsisRecIfUpdate (int handle,
                                          int mfn);

public static native int IsisRecIfUpdateEx (int handle,
                                            int beginmfn,
                                            int endmfn,
                                            int keeppending);

public static native int IsisRecIsoRead (int handle,
                                         int index);

public static native int IsisRecIsoWrite (int handle,
                                          int index);

public static native int IsisRecLeaderMap (int          handle,
                                           int          index,
                                           IsisRecLeader leader);

public static native int IsisRecLnk (int handle,
                                     int beginmfn,
                                     int endmfn);

public static native int IsisRecLockRecall (int handle,
                                            int index,
                                            int mfn,
                                            int tag,
```

```
                                        String password); public static native int IsisRecMerge
(int handle_from,                                   int index_from,
int handle_to,
                                    int index_to);

 public static native int IsisRecMfn (int handle,
                                    int index);

 public static native int IsisRecMfnChange (int handle,
                                        int index,
                                        int mfn);

 public static native int IsisRecNew (int handle,
                                    int index);

 public static native int IsisRecNewLock (int handle,
                                    int index);

 public static native int IsisRecNvf (int handle,
                                    int index);

 public static native int IsisRecRead (int handle,
                                    int index,
                                    int mfn);

 public static native int IsisRecReadLock (int handle,
                                    int index,
                                    int mfn);

 public static native int IsisRecShelfSize (int handle,
                                    int index,
                                    int mem);

 public static native int IsisRecSubField (int    handle,
                                    int    index,
                                    int    tag,
                                    int    fldocc,
                                    String subfield,
                                    String subfield_area[],
                                    int    areasize);

 public static native int IsisRecSubFieldEx (int    handle,
                                    int    index,
                                    int    tag,
                                    int    fldocc,
                                    String subfield,
                                    int    subfldocc,
                                    String subfield_area[],
                                    int    areasize);

 public static native int IsisRecUndelete (int handle,
                                    int index);

 public static native int IsisRecUnlock (int handle,
                                    int index);

 public static native int IsisRecUnlockForce (int handle,
                                    int index);
```

```
public static native int IsisRecUpdate (int    handle,
                                        int    index,
                                        String sparser);

public static native int IsisRecWrite (int handle,
                                       int index);

public static native int IsisRecWriteLock (int handle,
                                           int index);

public static native int IsisRecWriteUnlock (int handle,
                                             int index);


// 3.5. Space functions.


public static native int IsisSpaDb (int    handle,                                String dbname);

public static native int IsisSpaDelete (int handle);

public static native int IsisSpaFdt (int    handle,                               String fdtname);
public static native int IsisSpaFst (int    handle,
                                     String fstname);

public static native int IsisSpaGf (int    handle,
                                    String gizname);

public static native int IsisSpaHeaderMap (int          handle,
                                           IsisSpaHeader header);

public static native int IsisSpaIf (int    handle,
                                    String ifname);

public static native int IsisSpaIfCreate (int handle);

public static native int IsisSpaIsoDelim (int    handle,
                                          String recdelim,
                                          String fielddelim);

public static native int IsisSpaIsoIn (int    handle,
                                       String filename);

public static native int IsisSpaIsoOut (int    handle,
                                        String filename);

public static native int IsisSpaIsoOutCreate (int handle);

public static native int IsisSpaMf (int    handle,
                                    String mfname);

public static native int IsisSpaMfCreate (int handle);

public static native int IsisSpaMfUnlockForce (int handle);

public static native int IsisSpaName (int    handle,
                                      String sname);
```

```
 public static native int IsisSpaNew (int apphandle);

 public static native int IsisSpaPft (int    handle,
                                      String format);

public static native int IsisSpaRecDelim (int    handle,
                                          String begindelim,
                                          String enddelim);

 public static native int IsisSpaRecShelves (int handle,
                                             int max_mst);

 public static native int IsisSpaStw (int    handle,
                                      String stwname);

 public static native int IsisSpaTrmShelves (int handle,
                                             int max_trm);


 // 3.6. Search functions.

 public static native int IsisSrcHeaderMap (int          apphandle,
int          tsfnum,                                    int          searchnum,
                                            IsisSrcHeader header);

 public static native int IsisSrcLogFileFlush (int apphandle,
                                               int tsfnum);

 public static native int IsisSrcLogFileSave (int    apphandle,
                                              int    tsfnum,
                                              String filename);

 public static native int IsisSrcLogFileUse (int    apphandle,
                                             int    tsfnum,
                                             String filename);

 public static native int IsisSrcMfnMap (int       apphandle,
                                         int       tsfnum,
                                         int       searchnum,
                                         int       firstpos,
                                         int       lastpos,
                                         IsisSrcMfn mfn[]);

 public static native int IsisSrcRegExpMap (int          handle,
                                            String       expr,
                                            int          mfnb,
                                            int          mfne,
                                            String       areap,
                                            int          areasize);

 public static native int IsisSrcSearch (int          handle,
                                         int          tsfnum,
                                         String       express,
                                         IsisSrcHeader header);
```

```
public static native int IsisSrcSearchEx (int          handle,
          int          tsfnum,
                                          String     itable          String     express,
                                      IsisSrcHeader header);


 // 3.7. Term functions.


 public static native int IsisTrmMfnMap (int       handle,                                          int
index_trm,                                          int       firstpos,
                                          int       lastpos,
                                          IsisTrmMfn mfn[]);

 public static native int IsisTrmPostingMap (int          handle,
                                          int          index_trm,
                                          int          firstpos,
                                          int          lastpos,
                                          IsisTrmPosting posting[]);

 public static native int IsisTrmReadMap (int       handle,
                                          int       index_trm,
                                          IsisTrmRead key);

 public static native int IsisTrmReadNext (int       handle,
                                          int       index_trm,
                                          IsisTrmRead key);

 public static native int IsisTrmReadPrevious (int       handle,
                                          int       index_trm,
                                          String     prefix,
                                          IsisTrmRead key);

 public static native int IsisTrmShelfSize (int   handle,
                                          int   index_trm,
                                          int   mem);

 static
 { System.loadLibrary ("isis32"); }
}
```