IBM SPSS Statistics
Input/Output Module

IBM ®

Input/Output Module

IBM® SPSS® Statistics data files are binary files that contain the case data on which IBM SPSS Statistics operates and a dictionary describing the contents of the file. Many developers have successfully created applications that directly read and write IBM SPSS Statistics data files. Some of these developers have asked for a module to help them manipulate the rather complex format of IBM SPSS Statistics data files. The I/O Module documented here is designed to satisfy this need.

You can use the I/O Module to:

- Read and write IBM SPSS Statistics data files
- Set general file attributes, create variables
- Set values for variables
- Read cases
- Copy a dictionary
- Append cases to IBM SPSS Statistics data files
- Directly access data

Developers can call I/O Module procedures in client programs written in C, , and other programming languages. It is necessary to include the header file *spssdio.h*. The specific calling convention is __stdcall for both 32-bit and 64-bit Windows programs. The __stdcall conventions are compatible with FORTRAN, although calling I/O Module procedures is not specifically supported for FORTRAN programs.

This appendix outlines the steps for developing an application using the I/O Module procedures. It also contains a description of each procedure.

New I/O Module for Version 14.0

The I/O Module was completely rewritten for version 14.

- The new architecture should facilitate further development. However, much of the code is not used in the product itself and has not received as much testing as that in the predecessor module.

- An unintended but necessary limitation of the new module is that the spssOpenAppend function will not work correctly on compressed data files created by systems prior to release 14.

- To assist in the handling of non-western character sets, we are now using IBM's International Components for Unicode or ICU. As a result, the I/O Module depends on ICU runtime libraries, which are included on the CD-ROM.

- The I/O Module now uses the Microsoft Resident C Runtime. If the client application shares this run-time, it will also share the locale. As a result, any call to spssSetLocale will affect both the I/O Module and the client. Such a call is unnecessary if the client has already set the locale. When the module is loaded, it sets the locale to the system default.

- Prior to version 14.0.1, the name of the multiple response set specified for spssAddMultRespDefC or spssAddMultRespDefN was limited to 63 bytes, and the I/O Module automatically prepended a dollar sign. In the interest of consistency, the name is now limited to 64 bytes and must begin with a dollar sign. Also, the length of the set label was previously limited to 60 bytes. It may now be as long as a variable label, 255 bytes.

Using the I/O Module

The following sections list the sequence of procedures calls required to complete specific tasks with the I/O Module. See "I/O Module Procedure Reference" on p. 15 for detailed information about each procedure.

Interface and File Encoding

A new feature in version 16 is the option to represent text in a UTF-8 Unicode encoding rather than in the encoding of the current locale. If this option is chosen, all text (names, labels, values, and so on) communicated between the I/O Module and the

client application is represented as UTF-8, and the text in any output file will be represented as UTF-8. When in UTF-8 mode, the I/O Module can read files encoded in either mode and will perform the necessary transcoding to deliver UTF-8 text to the client. Conversely, when in code page mode, the I/O Module can read files encoded in either mode.

Data files created by release 15 and subsequent releases contain information about their encoding. When the I/O Module is operating in UTF-8 mode, it uses that information to perform the necessary transcoding. When the I/O Module is operating in code page mode, it will transcode from UTF-8 to the current local's encoding but will not transcode from one non-Unicode encoding to another. See the documentation of spssOpenAppend for some precautions when appending data to an open file.

Call spssSetInterfaceEncoding and spssGetInterfaceEncoding to set and get the interface encoding. Call spssGetFileEncoding and spssGetFileCodePage to get the encoding or code page of a specific file. Call spssIsCompatibleEncoding to determine whether the file and interface encoding are compatible.

When in UTF-8 mode the following apply:

- When retrieving string values--such as names, labels, or case values--values are returned as arrays of multibyte characters encoded in UTF-8. For example, in C, string values are returned as an array of char types encoded in UTF-8.

- When writing string values to an IBM SPSS Statistics data file, be sure that values are encoded in UTF-8. For example, in C, string values should be specified as an array of char types encoded in UTF-8.

- When creating an IBM SPSS Statistics data file, the file contains information specifying that the data are encoded in UTF-8. When viewing such a file in IBM SPSS Statistics, you should be working in unicode mode. You can specify unicode mode from the General tab on the Options dialog (Edit>Options), or by using the command syntax SET UNICODE=ON. Switching to unicode mode requires that there are no nonempty datasets open.

Example: Reading string values in UTF-8 mode

This example gets the variable names from an IBM SPSS Statistics data file and tests for the presence of a particular variable. It makes use of the Windows MultiByteToWideChar API to map the variable names (encoded in UTF-8) to UTF-16 (wide character) strings to allow comparison with a string literal.

```
#include "stdafx.h"
#include "spssdio.h"
#include <iostream>
#include "atlbase.h"
#include "atlstr.h"
using namespace std;
void func()
{
  int fH; /* file handle */
  int error; /* error code */
  int numV; /* number of variables */
  int *typesV; /* variable types */
  char **namesV; /* variable names */
  int res; /* Size, in characters, of buffer for variable name in UTF-16 */
  const wchar_t* name=L"résumé"; /* UTF-16 string literal of name to match */
  const size_t namesize=SPSS_MAX_VARNAME+1; /* UTF-16 variable name size*/
  wchar_t wcstring[namesize]; /* variable name in UTF-16 */
  double handlesV[100]; /* array of variable handles */

  error = spssSetInterfaceEncoding(SPSS_ENCODING_UTF8);
  error = spssOpenRead("mydata.sav",&fH);
  error = spssGetVarNames(fH, &numV, &namesV, &typesV);
  int i;
  for (i = 0; i < numV; ++i){
     error = spssGetVarHandle(fH, namesV[i], &handlesV[i]);
     if (error == SPSS_OK){
        res = MultiByteToWideChar(CP_UTF8, 0 , namesV[i], -1, wcstring, 0);
        MultiByteToWideChar(CP_UTF8, 0, namesV[i], -1, wcstring, res);
        if (!wcscmp(wcstring,name))
           cout << "Found match" << endl;
     }
     else ...
  }
  spssFreeVarNames(namesV, typesV, numV);
  error = spssCloseRead(fH);
}
```

For Visual Basic developers, the following is a Visual Basic version of the above example. It uses the spssGetVarInfo function to get the name and type of each variable,

one variable at a time. Also, it uses the Encoding class to handle conversions between UTF-8 and UTF-16.

```
Dim fH As Long 'file handle
Dim err As Integer 'error code
Dim varType As Integer 'variable type
Dim i As Integer
Dim numVars As Integer 'number of variables
Dim varName As String 'variable name
Dim varNameU As String 'variable name in Unicode (UTF-16)
Dim name As String = "résumé" 'UTF-16 string literal of name to match
Dim uniBytes() As Byte 'byte representation of variable name in UTF-16

varName = "".PadRight(SPSS_MAX_VARNAME + 1)
err = spssSetInterfaceEncoding(SPSS_ENCODING_UTF8)
err = spssOpenRead("mydata.sav", fH)
err = spssGetNumberofVariables(fH, numVars)
i = 0
Do While i < numVars
    err = spssGetVarInfo(fH, i, varName, varType)
    If (err = SPSS_OK) Then
        uniBytes = Encoding.Convert(Encoding.UTF8, Encoding.Unicode, _
                                    Encoding.Default.GetBytes(varName))
        varNameU = Encoding.Unicode.GetString(uniBytes).Trim()
        If (String.Compare(name, varNameU) = 0) Then
            Console.Write("Found match")
        End If
    End If
    i = i + 1
Loop
err = spssCloseRead(fH)
```

Example: Writing string values in UTF-8 mode

This example writes a new IBM SPSS Statistics data file in UTF-8 mode. It makes use of the Windows WideCharToMultiByte API to map a string literal in UTF-16 (wide

character) to the UTF-8 encoding required by IBM SPSS Statistics. For simplicity, it writes a file with a single variable and a single case value.

```
#include "stdafx.h"
#include "spssdio.h"
#include "atlbase.h"
#include "atlstr.h"
using namespace std;
void func()
{
  int fH; /* file handle */
  int error; /* error code */
  const wchar_t* val=L"männlich"; /* UTF-16 string to encode in UTF-8*/
  char varvalue[10]; /* character array for case value */
  double vH; /* variable handle */
  int res; /* Size, in bytes, of buffer for case value*/


  error = spssSetInterfaceEncoding(SPSS_ENCODING_UTF8);
  error = spssOpenWrite("mydata.sav", &fH);
  error = spssSetVarName(fH,"Geschlecht",SPSS_STRING(10));
  error = spssCommitHeader(fH);
  res = WideCharToMultiByte(CP_UTF8,0,val,-1,varvalue,0,NULL,NULL);
  WideCharToMultiByte(CP_UTF8,0,val,-1,varvalue,res,NULL,NULL);
  error = spssGetVarHandle(fH,"Geschlecht",&vH);
  error = spssSetValueChar(fH,vH,varvalue);
  error = spssCommitCaseRecord(fH);
  error = spssCloseWrite(fH);
}
```

For Visual Basic developers, the following is a Visual Basic version of the above example. It uses the Encoding class to handle conversions between UTF-16 and UTF-8.

```
Dim fH As Long 'file handle
Dim vH As Double 'variable handle
Dim err As Integer 'error code
Dim val As String = "männlich" 'String to encode in UTF-8
Dim utf8String As String 'String to pass to SPSS Statistics
Dim utf8Bytes() As Byte 'UTF-8 byte representation of string

spssSetInterfaceEncoding(SPSS_ENCODING_UTF8)
err = spssOpenWrite("mydata.sav", fH)
err = spssSetVarName(fH, "Geschlecht", 10)
err = spssCommitHeader(fH)
err = spssGetVarHandle(fH, "Geschlecht", vH)
utf8Bytes = Encoding.Convert(Encoding.Unicode, Encoding.UTF8, _
    Encoding.Unicode.GetBytes(val))
utf8String = Encoding.Default.GetString(utf8Bytes)
err = spssSetValueChar(fH, vH, utf8String)
err = spssCommitCaseRecord(fH)
err = spssCloseWrite(fH)
```

Writing IBM SPSS Statistics Data Files

The sequence of procedure calls to create IBM SPSS Statistics variables is as follows:

1. To open a physical file for output and to initialize internal data structures, call spssOpenWrite.

2. To set general file attributes, such as file label and compression, call spssSetIdString and spssSetCompression. These attributes may also be set anytime before the dictionary is committed (see step 7).

3. To create one or more variables, call spssSetVarName.

4. To set attributes of variables, such as output formats, variable and value labels, missing values, etc., call appropriate procedures, such as spssSetVarPrintFormat, spssSetVarLabel, spssSetVarNValueLabel, etc. Variable creation and attribute setting may be interleaved as long as no reference is made to a variable that has not yet been created.

5. (Optional) If you want to set variable sets, Trends date variables, or multiple response set information, call spssSetVariableSets, spssSetDateVariables, or spssSetMultRespDefs.

6. To set the case weight variable, call spssSetCaseWeightVar.

7. To commit the dictionary, call spssCommitHeader. Dictionary information may no longer be modified.

8. To prepare to set case data values, call spssGetVarHandle once for each variable and save the returned variable handles. A variable handle contains an index that allows data to be updated efficiently during case processing. While setting data values, variables must be referenced via their handles and not their names.

9. To set values of all variables for a case, call spssSetValueChar for string variables and spssSetValueNumeric for numeric ones. To write out the case, call spssCommitCaseRecord. Repeat from the beginning of this step until all cases are written.

10. To terminate file processing, call spssCloseWrite.

Utility procedures such as spssSysmisVal and any of the spssConvert procedures may be called at any time. They are useful primarily while setting case data values.

It is possible to construct complete cases in the form the cases would be written to an uncompressed data file and then present them to the I/O Module for output (which will take care of compression if necessary). Note that it is very easy to write out garbage this way. To use this approach, replace step 8 and step 9 with the following steps:

11. To obtain the size of an uncompressed case record in bytes, call spssGetCaseSize. Make sure that the size is what you think it should be. Allocate a buffer of that size.

12. Fill up the buffer with the correctly encoded numeric and string values, taking care of blank padding and doubleword alignment. To write the case, call spssWholeCaseOut. Repeat from the beginning of this step until all cases are written.

Copying a Dictionary

Developers can open a new file for output and initialize its dictionary from that of an existing file. The function, spssOpenWriteCopy, that implements this feature is a slight extension of spssOpenWrite. It is useful when the dictionary or data of an existing file is to be modified or all of its data is to be replaced. The typical sequence of operations is:

1. Call spssOpenWriteCopy (newFileName, oldFileName, ...) to open a new file initialized with a copy of the old file's dictionary.

2. Call spssOpenRead (oldFileName, ...) to access the old file's data.

Appending Cases to an Existing IBM SPSS Statistics Data File

To append cases, the existing data file must be compatible with the host system; that is, the system that originally created the file must use the same bit ordering and the same representation for the system-missing value as the host system. For example, a file created on a computer that uses high-order-first bit ordering (for example, Motorola) cannot be extended on an computer that uses low-order-first bit ordering (for example, Intel).

When appending cases, no changes are made to the dictionary other than the number of cases. The originating system and the creation date are not modified.

The sequence of procedure calls to append cases to an existing IBM SPSS Statistics data file is as follows:

1. To open a physical file and to initialize internal data structures, call spssOpenAppend.

2. To get general file attributes, such as file label, compression, and case weight, call spssGetIdString, spssGetCompression, and spssGetCaseWeightVar. To get the list of variable names and types, call spssGetVarNames, or call spssGetNumberofVariables and spssGetVarInfo if you are using Visual Basic. To get attributes of variables, such as output formats, variable and value labels, missing values, etc., call spssGetVarPrintFormat, spssGetVarLabel, spssGetVarNValueLabel(s), etc.

3. To set values of all variables for a case, call spssSetValueChar for string variables and spssSetValueNumeric for numeric variables. To append the case, call spssCommitCaseRecord. Repeat from the beginning of this step until all cases are written.

4. To terminate file processing, call spssCloseAppend.

Utility procedures such as spssSysmisVal and any of the spssConvert procedures may be called at any time. They are useful primarily while setting case data values.

For step 3, it is also possible to call spssWholeCaseOut to construct complete cases in the form in which the cases would be written to an uncompressed data file and then present them to the I/O Module for output (which will take care of compression if necessary). The same precaution should be taken as you write whole cases to a data file.

Reading IBM SPSS Statistics Data Files

The sequence of procedure calls to read IBM SPSS Statistics data files is much less restricted than the sequence of calls to write IBM SPSS Statistics data files. Cases, of course, must be read in sequence. However, calls that report file or variable attributes may be made anytime after the file is opened. A typical sequence of steps is:

1. To open a physical file for input and to initialize internal data structures, call spssOpenRead.

2. To get general file attributes, such as file label, compression, and case weight, call spssGetIdString, spssGetCompression, and spssGetCaseWeightVar. To get the list of variable names and types, call spssGetVarNames, or call spssGetNumberofVariables and spssGetVarInfo if you are using Visual Basic. To get attributes of variables, such as output formats, variable and value labels, missing values, etc., call spssGetVarPrintFormat, spssGetVarLabel, spssGetVarNValueLabel(s), etc.

3. (Optional) If you want to set variable sets, Trends date variables, or multiple response set information, call spssSetVariableSets, spssSetDateVariables, or spssSetMultRespDefs.

4. To find out the number of cases in the file, call spssGetNumberofCases.

5. To prepare to read case values, call spssGetVarHandle once for each variable whose values are of interest and save the returned variable handles. A variable handle contains an index that allows data to be retrieved efficiently during case processing. While retrieving data values, variables must be referenced via their handles and not their names.

6. To read the next case into the library's internal buffers, call spssReadCaseRecord. To get values of variables for a case, call spssGetValueChar for string variables and spssGetValueNumeric for numeric ones. Repeat from the beginning of this step until all cases are read.

7. To terminate file processing, call spssCloseRead.

Utility procedures such as spssSysmisVal and any of the spssConvert procedures may be called at any time. They are useful primarily while interpreting case data values. The spssFree... procedures should also be used where appropriate to free dynamically allocated data returned by the library.

Here, too, it is possible to receive from the I/O Module complete cases in the form in which the cases would appear in an uncompressed data file. Extracting data values from the case record is entirely up to the caller in this case. For this approach, replace step 5 and step 6 with the following steps:

8. To obtain the size of an uncompressed case record in bytes, call spssGetCaseSize. Allocate a buffer of that size.

9. To read the next case into your buffer, call spssWholeCaseIn. Extract the values you need from the buffer. Repeat from the beginning of this step until all cases are read.

Direct Access Input

The File I/O API supports direct access to the data in existing files. The basic mechanism is to call spssSeekNextCase, specifying a zero-origin case number before calling spssWholeCaseIn or spssReadCaseRecord. Note that direct reads from

compressed IBM SPSS Statistics data files require reading all of the data up to the requested case—that is, performance may not be sparkling when retrieving a few cases. Once an index of the cases has been constructed, performance is adequate.

Working with IBM SPSS Statistics Data Files

Variable Names and String Values

A user-definable IBM SPSS Statistics variable name must be valid in the current locale if working with a code page file. Variable names must always obey the following rules:

- The name must begin with a letter. The remaining characters may be any letter, any digit, a period, or the symbols @, #, _, or $.
- Variable names cannot end with a period. Names that end with an underscore should be avoided (to avoid name conflicts with variables automatically created by some procedures).
- The length of the name cannot exceed 64 bytes.
- Blanks and special characters (for example, !, ?, *) cannot be used.
- Each variable name must be unique; duplication is not allowed. Variable names are not case sensitive. The names *NEWVAR*, *NewVar*, and *newvar* are all considered identical.
- Reserved keywords (ALL, NE, EQ, TO, LE, LT, BY, OR, GT, AND, NOT, GE, and WITH) cannot be used.

If the names in a data file created in another locale are invalid in the current locale (for example, double-byte characters), the I/O Module will create acceptable names. These names are returned upon inquiry and can be used as legitimate parameters in procedures requiring variable names. The names in the data file will not be changed.

In the I/O Module, procedures that return variable names return them in upper case as null-terminated strings without any trailing blanks. Procedures that take variable names as input will accept mixed case and any number of trailing blanks without a problem. These procedures change everything to upper case and trim trailing blanks before using the variable names.

Similarly, procedures that return values of string variables return them as null-terminated strings whose lengths are equal to the lengths of the variables. Procedures

that take string variable values as input accept any number of trailing blanks and effectively trim the values to the variables' lengths before using them.

Accessing Variable and Value Labels

Beginning with version 7.5, the limit on the length of variable labels was increased from 120 to 256 bytes. There were two ways in which the spssGetVarLabel function could be modified to handle the longer labels. First, it could continue to return a maximum of 120 bytes for compatibility with existing applications. Second, it could return a maximum of SPSS_MAX_VARLABEL bytes for compatibility with new data files. The resolution was to continue to return a maximum of 120 bytes and to introduce a new function, spssGetVarLabelLong, which permits the client to specify the maximum number of bytes to return. In anticipation of possible future increases in the maximum width of value labels, two parallel functions, spssGetVarNValueLabelLong and spssGetVarCValueLabelLong, were added for retrieving the value labels of numeric and short string variables.

System-Missing Value

The special floating point value used to encode the system-missing value may differ from platform to platform, and the value encoded in a data file may differ from the one used on the host platform (one on which the application and the I/O Module are running). Files written through the I/O Module use the host system-missing value, which may be obtained by calling spssSysmisVal. For files being read using the I/O Module, data values having the system-missing value encoded in the file are converted to the host system-missing value; the system-missing value used in the data file is invisible to the user of the I/O Module.

Measurement Level, Column Width, and Alignment

Starting with release 8.0, IBM SPSS Statistics supports three additional variable attributes: measurement level, column width, and alignment. These attributes are not necessarily present IBM SPSS Statistics data files. However, when one attribute is recorded for a variable, all three must be recorded for every variable. Default values are assigned as necessary.

For example, if a new data file is being created and the measurement level attribute is explicitly set for one variable, default values will be assigned to measurement levels of all remaining variables, and column widths and alignments will be assigned to all variables. If no measurement level, column width, or alignment is assigned, the file will be written without values for any attribute.

There are six new file I/O API functions to access to these attributes: spssGetVarMeasureLevel, spssSetVarMeasureLevel, spssGetVarColumnWidth, spssSetVarColumnWidth, spssGetVarAlignment, and spssSetVarAlignment.

Support for Documents

IBM SPSS Statistics has a DOCUMENT command that can be used to store blocks of text in a data file. Until release 8.0, the I/O API had no support for documents—stored documents, if any, were discarded when opening an existing file, and there was no way to add documents to a new file. Starting with release 8.0, limited support for stored documents is provided that allows the user to retain existing documents.

When a file is opened for reading, its documents record is read and kept; if a file being written out has documents, they are stored in the dictionary. Since there is still no way to explicitly get or set documents, one may wonder how it is possible for an output file to acquire documents. The answer is, by using spssOpenWriteCopy to initialize a dictionary or by calling the spssCopyDocuments function to copy documents from one file to another. If an output file is created with spssOpenWriteCopy, the documents record of the file the dictionary is copied from is retained and written out when the dictionary is.

Coding Your Program

Any source file that references I/O Module procedures must include the header *spssdio.h*. The latter provides ANSI C prototypes for the I/O Module procedures and defines useful macros; it does not require any other headers to be included beyond what your program requires. To protect against name clashes, all I/O Module function names start with spss and all macro names are prefixed with SPSS_. In addition to the macros explicitly mentioned in the I/O Module procedures, *spssdio.h* defines macros for the maximum sizes of various data file objects that may help to make your program a little more readable:

SPSS_MAX_VARNAME   Variable name

SPSS_MAX_SHORTSTRING	Short string variable
SPSS_MAX_IDSTRING	File label string
SPSS_MAX_LONGSTRING	Long string variable
SPSS_MAX_VALLABEL	Value label
SPSS_MAX_VARLABEL	Variable label

Visual Basic Clients

The file *spssdio.vb* contains declarations of most of the API functions in a format that can be used in Visual Basic. The file also contains definitions of symbolic constants for all of the function return codes and the IBM SPSS Statistics format codes. Three comments are relevant to this file:

- It is necessary to have a knowledge of Chapter 26, "Calling Procedures in DLLs," in the *Microsoft Visual Basic Programmer's Guide*. Note that where the API function parameter should be an int, a Visual Basic application should use a long. Also, you should be careful to make string parameters suitably long before calling the API.

- Some functions, such as spssGetVarNames, are not compatible with being called from Visual Basic. The declarations of these functions are present only as comments.

- Only about 20% of the functions have actually been called from a working Visual Basic program. The inference is that some of the declarations are probably incorrect.

The function spssGetVarNames is a little difficult to call from languages other than C because it returns pointers to two vectors. BASIC and FORTRAN are not very well equipped to deal with pointers. Instead, use functions spssGetNumberofVariables and spssGetVarInfo, which enable the client program to access the same information in a little different way. Another function, spssHostSysmisVal, is provided as an alternative to spssSysmisVal to avoid returning a double on the stack.

Borland C++

Borland C++ users can use release 8.0.1 and later of *spssio32.dll* and the associated *spssdio.h*. They cannot, however, use the distributed *spssio32.lib*. It is necessary to

generate an import library from the distributed DLL using the *implib.exe* console application, which comes with the compiler using the following syntax:

implib -w spssio32.lib spssio32.dll

The -w switch suppresses almost 100 warnings, such as the following:

Warning duplicate symbol: spssCloseAppend

I/O Module Procedure Reference

The procedures are listed in alphabetical order.

spssAddFileAttribute

```
int spssAddFileAttribute(
   const int hFile,
   const char* attribName,
   const int attribSub,
   const char* attribText)
```

Description

This function adds a single datafile attribute. If the attribute already exists, it is replaced. The attribute name and its subscript are specified separately. The subscript is unit origin. If the attribute is not subscripted, the subscript must be specified as -1.

Parameter	Description
hfile	Handle to the data file
attribName	Name of the attribute. Not case sensitive.
attribSub	Unit origin subscript or -1
attribText	Text which used as the attribute's value

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	The file is read-only
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_INVALID_ATTRDEF	Missing name, missing text, or invalid subscript

SPSS_INVALID_ATTRNAME   Lexically invalid attribute name

spssAddMultRespDefC

> int spssAddMultRespDefC(int *handle*, const char **mrSetName*,
> const char **mrSetLabel*, int *isDichotomy*, const char **countedValue*,
> const char ***varNames*, int *numVars*)

Description

This function adds a multiple response set definition over short string variables to the dictionary.

Parameter	Description
handle	Handle to the data file.
mrSetName	Name of the multiple response set. A null-terminated string up to 64 bytes long that begins with a dollar sign and obeys the rules for a valid variable name. Case is immaterial.
mrSetLabel	Label for the multiple response set. A null-terminated string up to 256 bytes long. May be NULL or the empty string to indicate that no label is desired.
isDichotomy	Nonzero if the variables in the set are coded as dichotomies, zero otherwise.
countedValue	A null-terminated string containing the counted value. Necessary when isDichotomy is nonzero, in which case it must be 1–8 characters long, and ignored otherwise. May be NULL if isDichotomy is zero.
varNames	Array of null-terminated strings containing the names of the variables in the set. All variables in the list must be short strings. Case is immaterial.
numVars	Number of variables in the list (in varNames). Must be at least two.

Returns

If all goes well, adds the multiple response set to the dictionary and returns zero (SPSS_OK) or negative (a warning). Otherwise, returns a positive error code and does not add anything to the multiple response sets already defined, if any.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	The file is open for input or append
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_NO_VARIABLES	Fewer than two variables in list
SPSS_ EXC_STRVALUE	isDichotomy is nonzero and countedValue is NULL, empty, or longer than 8 characters
SPSS_INVALID_MRSETNAME	The multiple response set name is invalid
SPSS_DUP_MRSETNAME	The multiple response set name is a duplicate
SPSS_INVALID_MRSETDEF	Existing multiple response set definitions are invalid
SPSS_INVALID_VARNAME	One or more variable names in list are invalid
SPSS_VAR_NOTFOUND	One or more variables in list were not found in dictionary
SPSS_SHORTSTR_EXP	At least one variable in the list is numeric or long string
SPSS_NO_MEMORY	Insufficient memory to store the definition

spssAddMultRespDefExt

```
int spssAddMultRespDefExt(
    const int hFile,
    const spssMultRespDef* pSet)
```

Description

This function adds one multiple response set to the dictionary. The set is described in a struct which is defined in *spssdio.h*.

Parameter	Description
hFile	Handle to the data file
pSet	Pointer to the struct defining the set

The struct itself is defined as:

```
typedef struct spssMultRespDef_T
{
    char szMrSetName[SPSS_MAX_VARNAME+1];  /* Null-terminated MR set name */
    char szMrSetLabel[SPSS_MAX_VARLABEL+1];  /* Null-terminated set label */
    int qIsDichotomy;              /* Whether a multiple dichotomy set */
    int qIsNumeric;                /* Whether the counted value is numeric */
    int qUseCategoryLabels;        /* Whether to use var category labels */
    int qUseFirstVarLabel;         /* Whether using var label as set label */
    int Reserved[14];              /* Reserved for expansion */
    long nCountedValue;            /* Counted value if numeric */
   char* pszCountedValue;         /* Null-terminated counted value if string */
    char** ppszVarNames;           /* Vector of null-terminated var names */
    int nVariables;                /* Number of constituent variables */
} spssMultRespDef;
```

The items in the struct are as follows:

Item	Description
szMrSetName	Null-terminated name for the set. Up to 64 bytes. Must begin with "$".
szMrSetLabel	Null-terminated label for the set. Up to 256 bytes.
qIsDichotomy	True (non-zero) if this is a multiple dichotomy, i.e. an "MD" set.
qIsNumeric	True if the counted value is numeric. Applicable only to multiple dichotomies.
qUseCategoryLabels	True for multiple dichotomies for which the categories are to be labeled by the value labels corresponding to the counted value.
qUseFirstVarLabel	True for multiple dichotomies for which the label for the set is taken from the variable label of the first constituent variable.
nCountedValue	The counted value for numeric multiple dichotomies.
pszCountedValue	Pointer to the null-terminated counted value for character multiple dichotomies.
ppszVarNames	Pointer to a vector of null-terminated variable names.

nVariables		Number of variables in the set

When adding a set, the set name must be unique, and the variables must exist and be of the appropriate type - numeric or character depending on qIsNumeric.

Returns

The function returns SPSS_OK or an error value.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is invalid
SPSS_OPEN_RDMODE	The file is not open for output
SPSS_DICT_COMMIT	The dictionary has already been committed
SPSS_INVALID_MRSETNAME	Invalid name for the set
SPSS_INVALID_MRSETDEF	Invalid or inconsistent members of the definition struct.
SPSS_DUP_MRSETNAME	A set with the same name already exists

spssAddMultRespDefN

```
int spssAddMultRespDefN(int handle, const char *mrSetName,
const char *mrSetLabel, int isDichotomy, long countedValue,
const char **varNames, int numVars)
```

Description

This function adds a multiple response set definition over numeric variables to the dictionary.

Parameter	Description
handle	Handle to the data file.
mrSetName	Name of the multiple response set. A null-terminated string up to 64 bytes that begins with a dollar sign and obeys the rules for a valid variable name. Case is immaterial.

mrSetLabel	Label for the multiple response set. A null-terminated string up to 256 bytes long. May be NULL or the empty string to indicate no label is desired.
isDichotomy	Nonzero if the variables in the set are coded as dichotomies, zero otherwise.
countedValue	The counted value. Necessary when isDichotomy is nonzero and ignored otherwise. Note that the value is specified as a long int, not a double.
varNames	Array of null-terminated strings containing the names of the variables in the set. All variables in the list must be numeric. Case is immaterial.
numVars	Number of variables in the list (in varNames). Must be at least two.

Returns

If all goes well, adds the multiple response set to the dictionary and returns zero (SPSS_OK) or negative (a warning). Otherwise, returns a positive error code and does not add anything to the multiple response sets already defined, if any.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	The file is open for input or append
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_NO_VARIABLES	Fewer than two variables in list
SPSS_INVALID_MRSETNAME	The multiple response set name is invalid
SPSS_DUP_MRSETNAME	The multiple response set name is a duplicate
SPSS_INVALID_MRSETDEF	Existing multiple response set definitions are invalid
SPSS_INVALID_VARNAME	One or more variable names in list are invalid
SPSS_VAR_NOTFOUND	One or more variables in list were not found in dictionary
SPSS_NUME_EXP	At least one variable in the list is not numeric
SPSS_NO_MEMORY	Insufficient memory to store the definition

spssAddVarAttribute

```
int spssAddVarAttribute(
   const int hFile,
   const char* varName,
   const char* attribName,
   const int attribSub,
   const char* attribText)
```

Description

This function is analogous to spssAddFileAttribute, but it operates on a single variable's set of attributes. If the named attribute does not already exist, it is added to the set of attributes. If it does exist, the existing definition is replaced. If the attribute is not subscripted, the unit origin subscript is specified as -1.

Parameter	Description
hFile	Handle to the data file
varName	Name of the variable. Not case sensitive.
attribName	Name of the attribute. Not case sensitive.
attribSub	Unit origin attribute or -1
attribText	Text which used as the attribute's value

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_VAR_NOTFOUND	Named variable is not in the file
SPSS_OPEN_RDMODE	The file is read-only

SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_INVALID_ATTRDEF	Missing name, missing text, or invalid subscript
SPSS_INVALID_ATTRNAME	Lexically invalid attribute name

spssCloseAppend

int spssCloseAppend (int *handle*)

Description

This function closes the data file associated with *handle*, which must have been opened for appending cases using spssOpenAppend. The file handle *handle* becomes invalid and no further operations can be performed using it.

Parameter	Description
handle	Handle to the data file

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not appending, cases
SPSS_FILE_WERROR	File write error

Example

```
#include "spssdio.h"
void func()
{
  int fH;                    /* file handle */
  int error;                 /* error code  */
  ...
  error = spssOpenAppend("bank.sav", &fH);
  ...
  error = spssCloseAppend(fH);
  ...
  /* Handle fH is now invalid */
}
```

See also **spssOpenAppend**.

spssCloseRead

int spssCloseRead (int *handle*)

Description

This function closes the data file associated with *handle*, which must have been opened for reading using spssOpenRead. The file handle *handle* becomes invalid and no further operations can be performed using it.

Parameter	Description
handle	Handle to the data file

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_WRMODE	File is open for writing, not reading

Example

```
#include "spssdio.h"
void func()
{
  int fH;                   /* file handle */
  int error;                /* error code  */
  ...
  error = spssOpenRead("bank.sav", &fH);
  ...
  error = spssCloseRead(fH);
  ...
  /* Handle fH is now invalid */
}
```

See also **spssOpenRead**.

spssCloseWrite

int spssCloseWrite (int *handle*)

Description

This function closes the data file associated with *handle*, which must have been opened for writing using spssOpenWrite. The file handle *handle* becomes invalid and no further operations can be performed using it.

Parameter	Description
handle	Handle to the data file

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_NOTCOMMIT	Dictionary of the output file has not yet been written with spssCommitHeader

SPSS_FILE_WERROR	File write error

Example

See **spssSetValueNumeric**.

See also **spssOpenWrite**.

spssCommitCaseRecord

int spssCommitCaseRecord (int *handle*)

Description

This function writes a case to the data file specified by the *handle*. It must be called after setting the values of variables through spssSetValueNumeric and spssSetValueChar. Any variables left unset will get the system-missing value if they are numeric and all blanks if they are strings. Unless spssCommitCaseRecord is called, the case will not be written out.

Parameter	Description
handle	Handle to the data file

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_NOTCOMMIT	Dictionary of the output file has not yet been written with spssCommitHeader
SPSS_FILE_WERROR	File write error

Example

See **spssSetValueNumeric**.

See also **spssSetValueNumeric**, **spssSetValueChar**.

spssCommitHeader

int spssCommitHeader (int *handle*)

Description

This function writes the data dictionary to the data file associated with *handle*. Before any case data can be written, the dictionary must be committed; once the dictionary has been committed, no further changes can be made to it.

Parameter	Description
handle	Handle to the data file

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error.
SPSS_INVALID_HANDLE	The file handle is not valid.
SPSS_OPEN_RDMODE	File is open for reading, not writing.
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader.
SPSS_DICT_EMPTY	No variables defined in the dictionary.
SPSS_FILE_WERROR	File write error. In case of this error, the file associated with *handle* is closed and *handle* is no longer valid.
SPSS_NO_MEMORY	Insufficient memory.

SPSS_INTERNAL_VLABS   Internal data structures of the I/O Module are in-
                                       valid. This signals an error in the I/O Module.

Example

```
#include "spssdio.h"
void func()
{
  int fH;                    /* file handle */
  int error;                 /* error code  */
  ...
  error = spssOpenWrite("data.sav", &fH);
  ...
  /* Create some variables */
  error = spssSetVarName(fH, "AGE", SPSS_NUMERIC);
  ...
  /* Label variables -- Not required but useful */
  error = spssSetVarLabel(fH, "AGE", "Age of the Employee");
  ...
  /* Done with dictionary definition; commit dictionary */
  error = spssCommitHeader(fH);
  /* Handle errors... */
  ...
}
```

spssConvertDate

int spssConvertDate (int *day*, int *month*, int *year*, double **spssDate*)

Description

This function converts a Gregorian date expressed as day-month-year to the internal
date format. The time portion of the date variable is set to 0:00. To set the time portion
of the date variable to another value, use spssConvertTime and add the resulting value
to **spssDate*. Dates before October 15, 1582, are considered invalid.

Parameter	Description
day	Day of month (1–31)
month	Month (1–12)
year	Year in full (94 means 94 A.D.)
spssDate	Pointer to date in internal format

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_DATE	Invalid date

Example

```
#include "spssdio.h"
 void func()
 {
    int    fH;                  /* file handle      */
    int    error;               /* error code       */
    double vH;                  /* variable handle  */
    double sDate;               /* date             */
    double sTime;               /* time             */
    ...
    error = spssOpenWrite("data.sav", &fH);
    ...
    /* Create a numeric variable and set its print format
    ** to DATETIME28.4
    */
    error = spssSetVarName(fH, "TIMESTMP", SPSS_NUMERIC);
    ...
    error =
    spssSetVarPrintFormat(fH,"TIMESTMP",SPSS_FMT_DATE_TIME,4, 28);
    ...
    /* Get variable handle for TIMESTMP */
    error = spssGetVarHandle(fH, "TIMESTMP", &vH);
    ...
    /* Set value of TIMESTMP for first case to May 9, 1948,
    ** 10:30 AM. Do this by first using spssConvertDate to get
    ** a date value equal to May 9, 1948, 0:00 and adding to it
    ** a time value for 10:30:00.
    */
    error = spssConvertDate(9, 5, 1948, &sDate);
    ...
    /* Note that the seconds value is double, not int */
    error = spssConvertTime(0L, 10, 30, 0.0, &sTime);
    ...
    /* Set the value of the date variable */
    error = spssSetValueNumeric(fH, vH, sDate+sTime);
    ...
 }
```

See also **spssConvertTime**.

spssConvertSPSSDate

int spssConvertSPSSDate (int **day*, int **month*, int **year*, double *spssDate*)

Description

This function converts the date (as distinct from time) portion of a value in internal date format to Gregorian style.

Parameter	Description
day	Pointer to day of month value
month	Pointer to month value
year	Pointer to year value
spssDate	Date in internal format

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_DATE	The date value (*spssDate*) is negative

Example

```c
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int    fH;                  /* file handle      */
   int    error;               /* error code       */
   int    day, month, year;    /* date components  */
   int    hour, min;           /* time components  */
   long   jday;                /* Julian day       */
   double sec;                 /* seconds component*/
   double vH                   /* variable handle  */
   double sDate;               /* date+time        */
   ...
   error = spssOpenRead("myfile.sav", &fH);
   ...
   /* Get handle for TIMESTMP, a date variable */
   error = spssGetVarHandle(fH, "TIMESTMP" &vH);
   ...
   /* Read first case and print value of TIMESTMP */
   error = spssReadCaseRecord(fH);
   ...
   error = spssGetValueNumeric(fH, vH, &sDate);
   ...
   error = spssConvertSPSSDate(&day, &month, &year, sDate);
   ...
   /* We ignore jday, day number since Oct. 14, 1582 */
   error =
   spssConvertSPSSTime(&jday, &hour, &min, &sec, sDate);
   ...
   printf("Month/Day/Year: %d/%d/%d, H:M:S: %d:%d:%g\n",
          month, day, year, hour, min, sec);
   ...
}
```

spssConvertSPSSTime

int spssConvertSPSSTime
(long **day*, int **hour*, int **minute*, double **second*, double *spssTime*)

Description

This function breaks a value in internal date format into a day number (since October 14, 1582) plus the hour, minute, and second values. Note that the seconds value is stored in a double since it may have a fractional part.

Parameter	Description
day	Pointer to day count value (note that the value is long)
hour	Pointer to hour of day
minute	Pointer to minute of the hour
second	Pointer to second of the minute
spssTime	Date in internal format

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_TIME	The date value (*SpssTime*) is negative

Example

See **spssConvertSPSSDate**.

spssConvertTime

int spssConvertTime (long *day*, int *hour*, int *minute*, double *second*, double **spssTime*)

Description

This function converts a time given as day, hours, minutes, and seconds to the internal format. The day value is the number of days since October 14, 1582, and is typically zero, especially when this function is used in conjunction with spssConvertDate. Note that the seconds value is stored in a double since it may have a fractional part.

Parameter	Description
day	Day (non-negative; note that the value is long)
hour	Hour (0–23)
minute	Minute (0–59)
second	Seconds (non-negative and less than 60)
spssTime	Pointer to time in internal format

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_TIME	Invalid time

Example

See **spssConvertDate**.

See also **spssSetValueNumeric**.

spssCopyDocuments

int spssCopyDocuments (int *fromHandle*, int *toHandle*)

Description

This function copies stored documents, if any, from the file associated with *fromHandle* to that associated with *toHandle*. The latter must be open for output. If the target file already has documents, they are discarded. If the source file has no documents, the target will be set to have none, too.

Parameter	Description
fromHandle	Handle to the file documents are to be copied from.
toHandle	Handle to the file documents are to be copied to. Must be open for output.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	At least one handle is not valid
SPSS_OPEN_RDMODE	The target file is open for input or append
SPSS_DICT_COMMIT	spssCommitHeader has already been called for the target file

spssFreeAttributes

```
int spssFreeAttributes(
   char** attribNames,
   char** attribText,
   const int nAttributes)
```

Description

This function frees the memory dynamically allocated by either spssGetFileAttributes or spssGetVarAttributes.

Parameter	Description

attribNames	Pointer to the vector of attribute names
attribText	Pointer to the vector of text values
nAttributes	The number of elements in each vector

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_CANNOT_FREE	Program exception attempting to free the memory

spssFreeDateVariables

int spssFreeDateVariables (long* *dateInfo*)

Description

This function is called to return the memory allocated by spssGetDateVariables.

Parameter	Description
dateInfo	Vector of date variable indexes

Returns

Always returns SPSS_OK indicating success.

See also **spssGetDateVariables**.

spssFreeMultRespDefs

int spssFreeMultRespDefs(char **mrespDefs*)

Description

This function releases the memory which was acquired by spssGetMultRespDefs.

Parameter	Description
mrespDefs	ASCII string containing the definitions

Returns

The function always succeeds and always returns SPSS_OK.

See also **spssGetMultRespDefs**.

spssFreeMultRespDefStruct

```
int spssFreeMultRespDefStruct(
  spssMultRespDef* pSet)
```

Description

This function releases the memory acquired by spssGetMultRespDefByIndex. It has a single parameter, a pointer to the allocated struct.

Returns

The function returns SPSS_OK or an error code.

Error Code	Description
SPSS_OK	No error
SPSS_CANNOT_FREE	Cannot deallocate the memory, probably an invalid pointer

See also "spssGetMultRespDefByIndex" on page 52

spssFreeVarCValueLabels

```
int spssFreeVarCValueLabels (char **values, char **labels, int numLabels)
```

Description

This function frees the two arrays and the value and label strings allocated on the heap by spssGetVarCValueLabels.

Parameter	Description
values	Array of pointers to values returned by spssGetVarCValueLabels
labels	Array of pointers to labels returned by spssGetVarCValueLabels
numLabels	Number of values or labels returned by spssGetVarCValueLabels

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_CANNOT_FREE	Unable to free because arguments are illegal or inconsistent (for example, negative *numLabels*)

Example

See **spssGetVarNValueLabels**.

See also **spssFreeVarCValueLabels**.

spssFreeVariableSets

int spssFreeVariableSets (char *varSets)

Description

This function is called to return the memory allocated by spssGetVariableSets.

Parameter	Description
varSets	The string defining the variable sets

Returns

Always returns SPSS_OK indicating success.

See also **spssGetVariableSets**.

spssFreeVarNValueLabels

int spssFreeVarNValueLabels (double *values*, char **labels*, int *numLabels*)

Description

This function frees the two arrays and the label strings allocated on the heap by spssGetVarNValueLabels.

Parameter	Description
values	Array of values returned by spssGetVarNValueLabels
labels	Array of pointers to labels returned by spssGetVarNValueLabels
numLabels	Number of values or labels returned by spssGetVarNValueLabels

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_CANNOT_FREE	Unable to free because arguments are illegal or inconsistent (for example, negative *numLabels*)

Example

See **spssGetVarNValueLabels**.

See also **spssFreeVarCValueLabels**.

spssFreeVarNames

int spssFreeVarNames (char **varNames, int *varTypes, int *numVars*)

Description

This function frees the two arrays and the name strings allocated on the heap by spssGetVarNames.

Parameter	Description
varNames	Array of pointers to names returned by spssGetVarNames
varTypes	Array of variable types returned by spssGetVarNames
numVars	Number of variables returned by spssGetVarNames

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_CANNOT_FREE	Unable to free because arguments are illegal or inconsistent (for example, negative *numVars*)

Example

See **spssGetVarNames**.

spssGetCaseSize

int spssGetCaseSize (int *handle*, long *caseSize)

Description

This function reports the size of a raw case record for the file associated with *handle*. The case size is reported in bytes and is meant to be used in conjunction with the low-level case input/output procedures spssWholeCaseIn and spssWholeCaseOut.

Parameter	Description
handle	Handle to the data file
caseSize	Pointer to size of case in bytes

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_DICT_NOTCOMMIT	The file is open for output, and the dictionary has not yet been written with spssCommitHeader

Example

See **spssWholeCaseIn**.

See also **spssWholeCaseIn**, **spssWholeCaseOut**.

spssGetCaseWeightVar

int spssGetCaseWeightVar (int *handle*, const char **varName*)

Description

This function reports the name of the case weight variable. The name is copied to the buffer pointed to by *varName* as a null-terminated string. Since a variable name can be up to 64 bytes in length, the size of the buffer must be at least 65.

Parameter	Description
handle	Handle to the data file
varName	Pointer to the buffer to hold name of the case weight variable

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error.
SPSS_NO_CASEWGT	A case weight variable has not been defined for this file (warning).
SPSS_INVALID_HANDLE	The file handle is not valid.
SPSS_INVALID_CASEWGT	The given case weight variable is invalid. This error signals an internal problem in the implementation of the I/O Module and should never occur.

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int fH;                   /* file handle        */
   int error;                /* error code         */
    char caseWeight[9];      /* case weight variable */
   ...
   error = spssOpenRead("bank.sav", &fH);
   ...
   /* Get and print the case weight variable of this file */
   error = spssGetCaseWeightVar(fH, caseWeight);
   if (error == SPSS_NO_CASEWGT)
    printf("The file is unweighted.\n");
   else if (error == SPSS_OK)
    printf("The case weight variable is: %s\n", caseWeight);
   else /* Handle error */
    ...
}
```

spssGetCompression

int spssGetCompression (int *handle*, int **compSwitch*)

Description

This function reports the compression attribute of a data file.

Parameter	Description
handle	Handle to the data file.
compSwitch	Pointer to compression attribute. Upon return, **compSwitch* is 1 if the file is compressed; 0 otherwise.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int fH;                  /* file handle        */
   int error;               /* error code         */
   int compSwitch;          /* compression switch */
   ...
   error = spssOpenRead("bank.sav", &fH);
   ...
   /* Print whether the data file is compressed. */
   error = spssGetCompression(fH, &compSwitch);
   if (error == SPSS_OK)
   {
    printf("File is ");
    if (compSwitch)
     printf("compressed.\n");
    else
     printf("uncompressed.\n");
   }
}
```

spssGetDateVariables

int spssGetDateVariables (int *handle*, int **numofElements*, long ***dateInfo*)

Description

This function reports the Forecasting (Trends) date variable information, if any, in IBM SPSS Statistics data files. It places the information in a dynamically allocated long array, sets **numofElements* to the number of elements in the array, and sets **dateInfo* to point to the array. The caller is expected to free the array by calling spssFreeDateVariables when it is no longer needed. The variable information is copied directly from record 7, subtype 3. Its first six elements comprise the "fixed" information, followed by a sequence of one or more three-element groups.

Parameter	Description
handle	Handle to the data file
numofElements	Number of elements in allocated array
dateInfo	Pointer to first element of the allocated array

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_DATEINFO	There is no Trends date variable information in the file (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "spssdio.h"
void func()
{
   int  fH;                 /* file handle              */
   int  numD;               /* number of elements       */
   long *dateInfo;          /* pointer to date variable info. */
   ...
   error = spssOpenRead("bank.sav", &fH);
   ...
   /* Get & print TRENDS date variables info. */
   error = spssGetDateVariables(fH, &numD, &dateInfo);
   if (error == SPSS_NO_DATEINFO)
    printf("No TRENDS information.\n");
   else if (error == SPSS_OK)
   {
    if (numD < 6 || numD%3 != 0)
    {
     /* Should never happen */
     printf("Date info format error.\n");
     free(dateInfo);
     return;
    }
    /*Print the first six elements followed by groups of three */
    ...
    /* Remember to free array */
    spssFreeDateVariables(dateInfo);
   }
   ...
}
```

See also **spssSetDateVariables, spssFreeDateVariables**.

spssGetDEWFirst

int spssGetDEWFirst (const int *handle*, void **pData*, const long *maxData*, long **nData*)

Description

The client can retrieve DEW information (file information that is private to the Data Entry product) from a file in whatever increments are convenient. The first such increment is retrieved by calling spssGetDEWFirst, and subsequent segments are retrieved by calling spssGetDEWNext as many times as necessary. As with spssGetDEWInfo, spssGetDEWFirst will return SPSS_NO_DEW if the file was written with a byte order that is the reverse of that of the host.

Parameter	Description
handle	Handle to the data file
pData	Returned as data from the file
maxData	Maximum bytes to return
nData	Returned as number of bytes returned

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_DEW	File contains no DEW information (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_FILE_BADTEMP	Error accessing the temporary file

See also **spssGetDEWInfo**, **spssGetDEWNext**.

spssGetDEWGUID

int spssGetDEWGUID (const int handle, char* asciiGUID)

Description

Data Entry for Windows maintains a GUID in character form as a uniqueness indicator. Two files have identical dictionaries and DEW information if they have the same GUID. Note that the spssOpenWriteCopy function will not copy the source file's GUID. spssGetDEWGUID allows the client to read a file's GUID, if any. The client supplies a 257 byte string in which the null-terminated GUID is returned.

Parameter	Description
handle	Handle to the data file
asciiGUID	Returned as the file's GUID in character form or a null string if the file contains no GUID

Returns

The GUID is returned as a null-terminated string in parameter asciiGUID. If the file does not contain a GUID (and most do not), a null string is returned. When a null string is returned, the function result will still be SPSS_OK.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid

See also **spssSetDEWGUID**.

spssGetDEWInfo

int spssGetDEWInfo (const int *handle*, long **pLength*, long **pHashTotal*)

Description

This function can be called before actually retrieving DEW information (file information that is private to the Data Entry product) from a file, to obtain some attributes of that information—specifically its length in bytes and its hash total. The hash total is, by convention, contained in the last four bytes to be written. Because it is not cognizant of the structure of the DEW information, the I/O Module is unable to correct the byte order of numeric information generated on a foreign host. As a result,

the DEW information is discarded if the file has a byte order that is the reverse of that of the host, and calls to spssGetDEWInfo will return SPSS_NO_DEW.

Parameter	Description
handle	Handle to the data file
pLength	Returned as the length in bytes
pHashTotal	Returned as the hash total

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_NO_DEW	File contains no DEW information (warning)

spssGetDEWNext

int spssGetDEWNext (const int *handle*, void **pData*, const long *maxData*, long **nData*)

Description

The client can retrieve DEW information (file information that is private to the Data Entry product) from a file in whatever increments are convenient. The first such increment is retrieved by calling spssGetDEWFirst, and subsequent segments are retrieved by calling spssGetDEWNext as many times as necessary. As with spssGetDEWInfo, spssGetDEWFirst will return SPSS_NO_DEW if the file was written with a byte order that is the reverse of that of the host.

Parameter	Description
handle	Handle to the data file
pData	Returned as data from the file
maxData	Maximum bytes to return

nData	Returned as number of bytes returned

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_DEW_NOFIRST	spssGetDEWFirst was never called
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_FILE_BADTEMP	Error accessing the temporary file

See also **spssGetDEWInfo**, **spssGetDEWFirst**.

spssGetEstimatedNofCases

int spssGetEstimatedNofCases(const int *handle*, long **caseCount*)

Description

Although not strictly required for direct access input, this function helps in reading IBM SPSS Statistics data files from releases earlier than 6.0. Some of these data files did not contain number of cases information, and spssGetNumberofCases will return –1 cases. This function will return a precise number for uncompressed files and an estimate (based on overall file size) for compressed files. It cannot be used on files open for appending data.

Parameter	Description
handle	Handle to the data file
caseCount	Returned as estimated *n* of cases

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_WRMODE	The file is open for writing, not reading
SPSS_FILE_RERROR	Error reading the file

See also **spssGetNumberofCases**.

spssGetFileAttributes

```
int spssGetFileAttributes(
    const int hFile,
    char*** attribNames,
    char*** attribText,
    int* nAttributes)
```

Description

This function returns all the datafile attributes. It allocates the memory necessary to hold the attribute names and values. For subscripted attributes, the names include the unit origin subscripts enclosed in square brackets, for example Prerequisite[11]. The acquired memory must be released by calling spssFreeAttributes.

Parameter	Description
hFile	handle to the data file
attribNames	Returned as a pointer to a vector of attribute names
attribText	Returned as a pointer to a vector of attribute values
nAttributes	Returned as the number of element in each vector

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error

SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_NO_MEMORY	Insufficient memory for the vectors

spssGetFileCodePage

```
int spssGetFileEncoding(
   const int hFile,
   int* nCodePage)
```

Description

This function provides the Windows code page number of the encoding applicable to a file. For instance, the Windows code page for ISO-8859-1 is 28591. Note that the Windows code page for UTF-8 is 65001.

Parameter	Description
hFile	Handle to the file
nCodePage	Returned as the code page of the file

Returns

The function returns SPSS_OK or an error value:

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is invalid
SPSS_INCOMPATIBLE_DICT	There is no Windows code page equivalent for the file's encoding

spssGetFileEncoding

```
int spssGetFileEncoding(
   const int hFile,
   char* pszEncoding)
```

Description

This function obtains the encoding applicable to a file. The encoding is returned as an IANA encoding name, such as ISO-8859-1. The maximum length of the returned string is SPSS_MAX_ENCODING plus a null terminator.

Parameter	Description
hFile	Handle to the file
pszEncoding	Returned as the encoding of the file

Returns

The function returns SPSS_OK or an error value:

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is invalid

spssGetIdString

int spssGetIdString (int *handle*, char **id*)

Description

This function copies the file label of the IBM SPSS Statistics data file associated with *handle* into the buffer pointed to by *id*. The label is at most 64 characters long and null-terminated. Thus, the size of the buffer should be at least 65. If an input data file is associated with the handle, the label will be exactly 64 characters long, padded with blanks as necessary.

Parameter	Description
handle	Handle to the data file
id	File label buffer

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int fH;                    /* file handle */
   int error;                 /* error code  */
   char id[65];               /* file label  */
   ...
   error = spssOpenRead("bank.sav", &fH);
   ...
   error = spssGetIdString(fH, id);
   if (error == SPSS_OK)
    printf("File label: %s\n", id);
   ...
}
```

spssGetInterfaceEncoding

int spssGetInterfaceEncoding()

Description

This function returns the current interface encoding.

Returns

The function returns SPSS_ENCODING_CODEPAGE or SPSS_ENCODING_UTF8.

spssGetMultRespCount

```
int spssGetMultRespCount(
   const int hFile,
   int* nSets)
```

Description

This function obtains a count of the number of multiple response sets stored in the dictionary.

Parameter	Description
hFile	Handle to the data file
nSets	Returned as the number of sets

Returns

The function returns SPSS_OK or an error value:

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is invalid
SPSS_OPEN_WRMODE	The file is not open for input

spssGetMultRespDefByIndex

```
int spssGetMultRespDefByIndex(
    const int hFile,
    const int iSet,
    spssMultRespDef** ppSet)
```

Description

This function obtains a description of a single multiple response set. The set is specified via a zero origin index, and the description is returned in a struct for which the memory is allocated by the function.

Parameter	Description
hFile	Handle to the data set
iSet	Zero origin index of the set
ppSet	Returned as a pointer to the set's description

For information on the set description struct, see "spssAddMultRespDefExt" on page 17. The memory for the struct must be freed by calling spssFreeMultRespDefStruct.

Returns

The function returns SPSS_OK or an error code.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is invalid
SPSS_OPEN_WRMODE	The file is not open for input
SPSS_INVALID_MRSETINDEX	The index is out of range
SPSS_NO_MEMORY	Insufficient memory to allocate the struct

spssGetMultRespDefs

int spssGetMultRespDefs (const int *handle*, char **mrespDefs*)

Parameter	Description
handle	Handle to the data file
mrespDefs	Returned as a pointer to a string

Description

This function retrieves the multiple response set definitions from IBM SPSS Statistics data files. The definitions are stored as a null-terminated code page or UTF-8 string based on whether the spssGetInterfaceEncoding() type is SPSS_ENCODING_CODEPAGE or SPSS_ENCODING_UTF8. The memory allocated by this function to contain the string must be freed by calling spssFreeMultRespDefs. If the file contains no multiple response definitions, *mrespDefs* is set to NULL, and the function returns the warning code SPSS_NO_MULTRESP.

For multiple category sets the string contains the following:
$setname=C {label length} {label} {variable list}

For multiple dichotomy sets, the string contains the following:
$setname=D{value length} {counted value} {label length} {label} {variable list}

- All multiple multiple category and multiple dichotomy sets in the data file are returned as single string, with a newline character (\n) between each set.

- All multiple response set names begin with a dollar sign and follow variable naming rules.

- For mulitple dichotomy sets, there is no space between the "D" and the integer that represents the length of the counted value.

- If there is no label for the set, the label length is 0, and there is a single blank space for the label. (So there are two blank spaces between the label length value of 0 and the first variable name.)

For example:

```
$mcset=C 21 Multiple Category Set mcvar1 mcvar2 mcvar3 mcvar4 \n
$mdset1=D1 1 22 Multiple Dichotomy Set  mdvar1 mdvar2 mdvar3 mdvar4 \n
$mdset2=D3 Yes 0  mdvar5 mdvar6 mdvar7
```

Note: For multiple dichotomy sets that use counted values as category labels (CATEGORYLABELS=COUNTEDVALUES in SPSS Statistics command syntax) or the variable label of the first set variable as the set label (LABELSOURCE=VARLABEL in SPSS Statistics command syntax), use the method spssGetMultRespDefsEx.

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_MULTRESP	No definitions on the file (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_NO_MEMORY	Insufficient memory to contain the string

See also **spssFreeMultRespDefs**, **spssGetMultRespDefsEx**.

spssGetMultRespDefsEx

int spssGetMultRespDefsEx (const int *handle*, char **mrespDefs*)

Parameter	Description
handle	Handle to the data file

       mrespDefs   Returned as a pointer to a string

Description

This function retrieves the multiple response set definitions for from IBM SPSS Statistics data files for "extended" multiple dichotomy sets. The definitions are stored as a null-terminated code page or UTF-8 string based on whether the spssGetInterfaceEncoding() type is SPSS_ENCODING_CODEPAGE or SPSS_ENCODING_UTF8. The memory allocated by this function to contain the string must be freed by calling spssFreeMultRespDefs. If the file contains no multiple response definitions, **mrespDefs* is set to NULL, and the function returns the warning code SPSS_NO_MULTRESP.

    An "extended" multiple dichotomy is a set that uses counted values as category labels (CATEGORYLABELS=COUNTEDVALUES in SPSS Statistics command syntax) or the variable label of the first set variable as the set label (LABELSOURCE=VARLABEL in SPSS Statistics command syntax).

The string contains the following:
$setname=E {flag1}[flag2] {value length} {counted value} {label length} {label} {variable list}

- All extended dichotomy sets in the data file are returned as single string, with a newline character (\n) between each set.
- All multiple response set names begin with a dollar sign and follow variable naming rules.
- flag1 always has a value of 1 and indicates that counted values are used as category labels.
- flag2 has a value of 1 if the variable label of the first variable in the set is used as the label; otherwise flag2 is not included. There is no space between flag1 and flag2.
- If there is no label for the set, the label length is 0. The label length is always 0 if flag2 is present (and set to 1). If the label length is 0, there is a single blank space for the label. (So there are two blank spaces between the label length value of 0 and the first variable name.)

For example:

$meset=E 11 1 1 0 mevar1 mevar2 mevar3 \n
$meset=E 1 3 Yes 38 Enhanced set with user specified label mevar4 mevar5 mevar6

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_MULTRESP	No definitions on the file (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_NO_MEMORY	Insufficient memory to contain the string

See also **spssFreeMultRespDefs**, **spssGetMultRespDef**.

spssGetNumberofCases

int spssGetNumberofCases (int *handle*, long **numofCases*)

Description

This function reports the number of cases present in a data file open for reading.

Parameter	Description
handle	Handle to the data file
numofCases	Pointer to number of cases

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_WRMODE	File is open for writing, not reading

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int fH;                  /* file handle      */
   int error;               /* error code       */
   long count;              /* Number of cases  */
   ...
   error = spssOpenRead("bank.sav", &fH);
   ...
   /* Get & print the number of cases present in the file. */
   error = spssGetNumberofCases(fH, &count);
   if (error == SPSS_OK)
    printf("Number of cases: %ld\n");
   ...
}
```

spssGetNumberofVariables

int spssGetNumberofVariables (int *handle*, long **numVars*)

Description

This function reports the number of variables present in a data file.

Parameter	Description
handle	Handle to the data file
numVars	Pointer to number of variables

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_DICT_NOTCOMMIT	Dictionary has not been committed
SPSS_INVALID_FILE	Data file contains no variables

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
  int fH;                    /* file handle        */
  int error;                 /* error code         */
  long count;                /* Number of variables*/
  ...
  error = spssOpenRead("bank.sav", &fH);
  ...
  /* Get & print the number of variables present in the file. */
  error = spssGetNumberofVariables(fH, &count);
  if (error == SPSS_OK)
   printf("Number of variables: %ld\n");
  ...
}
```

spssGetReleaseInfo

int spssGetReleaseInfo (int *handle*, int *relinfo[]*)

Description

This function reports release- and machine-specific information about the file associated with *handle*. The information consists of an array of eight int values copied from record type 7, subtype 3 of the file, and is useful primarily for debugging. The array elements are, in order, release number (index 0), release subnumber (1), special release identifier number (2), machine code (3), floating-point representation code (4), compression scheme code (5), big/little-endian code (6), and character representation code (7).

Parameter	Description
handle	Handle to the data file.
relinfo	Array of int in which release- and machine-specific data will be stored. This array must have at least eight elements.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values (with one exception noted below).

Error Code	Description
SPSS_OK	No error.
SPSS_INVALID_HANDLE	The file handle is not valid.
SPSS_NO_TYPE73	There is no type 7, subtype 3 record present. This code should be regarded as a warning even though it is positive. Files without this record are valid.

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int fH;                  /* file handle       */
   int error;               /* error code        */
   int relInfo[8];          /* release info      */
   ...
   error = spssOpenRead("bank.sav", &fH);
   ...
   /* Get & print release and machine-specific info. */
   error = spssGetReleaseInfo(fH, relInfo);
   if (error == SPSS_OK)
   {
    printf("Release & machine information:\n");
    int i;
    for (i = 0; i < 8; ++i)
     printf("  Element %d: %d\n", i, relInfo[i]);
   }
   ...
}
```

spssGetSystemString

int spssGetSystemString (int *handle*, char **sysName*)

Description

This function returns the name of the system under which the file was created. It is a 40-byte blank-padded character field corresponding to the last 40 bytes of record type 1. Thus, in order to accommodate the information, the parameter *sysName* must be at least 41 bytes in length plus the terminating null character.

Parameter	Description
handle	Handle to the data file

sysName	The originating system name

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int fH;                      /* file handle        */
   int error;                   /* error code         */
   char sysName[41];            /* orignating system  */
   ...
   error = spssOpenRead("bank.sav", &fH);
   ...
   error = spssGetIdString(fH, sysName);
   if (error == SPSS_OK)
    printf("Originating System: %s\n", sysName);
   ...
}
```

spssGetTextInfo

int spssGetTextInfo (int *handle*, char **textInfo*)

Description

This function places the text data created by TextSmart as a null-terminated string in the user-supplied buffer *textInfo*. The buffer is assumed to be at least 256 characters long; the text data may be up to 255 characters long. If text data are not present in the file, the first character in *textInfo* is set to NULL.

Parameter	Description
handle	Handle to the data file

textInfo	Buffer for text data

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid

spssGetTimeStamp

int spssGetTimeStamp (int *handle*, char **fileDate,* char **fileTime*)

Description

This function returns the creation date of the file as recorded in the file itself. The creation date is a null-terminated 9-byte character field in dd mmm yy format (27 Feb 96), and the receiving field must be at least 10 bytes in length. The creation time is a null-terminated 8-byte character field in hh:mm:ss format (13:12:15), and the receiving field must be at least 9 bytes in length.

Parameter	Description
handle	Handle to the data file
fileDate	File creation date
fileTime	File creation time

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid

spssGetValueChar

int spssGetValueChar (int *handle*, double *varHandle*, char **value*, int *valueSize*)

Description

This function gets the value of a string variable for the current case, which is the case read by the most recent call to spssReadCaseRecord. The value is returned as a null-terminated string in the caller-provided buffer *value*; the length of the string is the length of the string variable. The argument *valueSize* is the allocated size of the buffer *value*, which must be at least the length of the variable plus 1.

Parameter	Description
handle	Handle to the data file
varHandle	Handle of the variable
value	Buffer for the value of the string variable
valueSize	Size of *value*

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error.
SPSS_INVALID_HANDLE	The file handle is not valid.
SPSS_OPEN_WRMODE	File is open for writing, not reading.
SPSS_INVALID_CASE	Current case is not valid. This may be because no spssReadCaseRecord calls have been made yet or because the most recent call failed with error or encountered the end of file.
SPSS_STR_EXP	Variable associated with the handle is numeric.
SPSS_BUFFER_SHORT	Buffer *value* is too short to hold the value.

Example

```c
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int  fH;                   /* file handle         */
   int  error;                /* error code          */
   int  numV;                 /* number of variables */
   int  *typesV;              /* variable types      */
   char **namesV;             /* variable names      */
   double handlesV[100]; /* assume no more than 100 variables */
   char cValue[256];     /* long enough for any string variable */
   long nCases;               /* number of cases     */
   long casesPrint;           /* number of cases to print */
   long case;                 /* case index */
   double nValue;             /* numeric value       */
   int i;                     /* variable index      */
   ...
   error = spssOpenRead("bank.sav", &fH);
   ...
   /* Get variable names and types */
   error = spssGetVarNames(fH, &numV, &namesV, &typesV);
   ...
   if (numV > 100)
   {
    printf("Too many variables; increase program capacity.\n");
    spssFreeVarNames(namesV, typesV, numV);
    return;
   }
   /* Get & store variable handles */
   for (i = 0; i < numV; ++i)
   {
    error = spssGetVarHandle(fH, namesV[i], &handlesV[i]);
    if (error != SPSS_OK) ...
   }
   /* Get the number of cases */
   error = spssGetNumberofCases(fH, &nCases);
   ...
   /* Print at most the first ten cases */
   casesPrint = (nCases < 10) ? nCases : 10;
   for (case = 1; case <= casesPrint; ++case)
   {
    error = spssReadCaseRecord(fH);
    ...
    printf("Case %ld\n", case);
    for (i = 0; i < numV; ++I)
    {
     if (typesV[i] == 0)
     {
      /* Numeric */
      error = spssGetValueNumeric(fH, handlesV[i], &nValue);
      if (error == SPSS_OK)
       printf("    %ld\n", nValue);
      else ...
     }
     else
     {
      /* String */
      error = spssGetValueChar(fH, handlesV[i], cValue, 256);
      if (error == SPSS_OK)
       printf("    %s\n", cValue);
      else ...
     }
    }
   }
}
```

```
        /* Free the variable names & types */
         spssFreeVarNames(namesV, typesV, numV);
}
```

spssGetValueNumeric

int spssGetValueNumeric (int *handle*, double *varHandle*, double **value*)

Description

This function gets the value of a numeric variable for the current case, which is the case read by the most recent call to spssReadCaseRecord.

Parameter	Description
handle	Handle to the data file
varHandle	Handle to the variable
value	Pointer to the value of the numeric variable

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error.
SPSS_INVALID_HANDLE	The file handle is not valid.
SPSS_OPEN_WRMODE	File is open for writing, not reading.
SPSS_INVALID_CASE	Current case is not valid. This may be because no spssReadCaseRecord calls have been made yet or because the most recent call failed with error or encountered the end of file.
SPSS_NUME_EXP	Variable associated with the handle is not numeric.

Example

See **spssGetValueChar**.

spssGetVarAttributes

```
int spssGetVarAttributes(
    const int hFile,
    const char* varName,
    char*** attribNames,
    char*** attribText,
    int* nAttributes)
```

Description

This function is analogous to spssGetFileAttributes. It returns all the attributes for a single variable.

Parameter	Description
hFile	Handle to the data file
varName	The name of the variable
attribNames	Returned as a pointer to a vector of attribute names
attribText	Returned as a pointer to a vector of attribute values
nAttributes	Returned as the number of element in each vector

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_VAR_NOTFOUND	Named variable is not in the file
SPSS_NO_MEMORY	Insufficient memory for the vectors

spssGetVarAlignment

int spssGetVarAlignment (int *handle*, const char **varName*, int **alignment*)

Description

This function reports the value of the alignment attribute of a variable.

Parameter	Description
handle	Handle to the data file.
varName	Variable name.
alignment	Pointer to alignment. Set to SPSS_ALIGN_LEFT, SPSS_ALIGN_RIGHT, or SPSS_ALIGN_CENTER.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist

spssGetVarCMissingValues

```
int spssGetVarCMissingValues
(int handle, const char *varName, int *missingFormat,
char *missingVal1, char *missingVal2, char *missingVal3)
```

Description

This function reports the missing values of a short string variable. The value of *missingFormat will be in the range 0–3, indicating the number of missing values. The appropriate number of missing values is copied to the buffers *missingVal1*, *missingVal2*, and *missingVal3*. The lengths of the null-terminated missing value strings will be the length of the short string variable in question. Since the latter can be at most 8 characters long, 9-character buffers are adequate for any short string variable.

Parameter	Description
handle	Handle to the data file
varName	Variable name
missingFormat	Pointer to missing value format code
missingVal1	Buffer for first missing value
missingVal2	Buffer for second missing value
missingVal3	Buffer for third missing value

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_STR_EXP	The variable is numeric
SPSS_SHORTSTR_EXP	The variable is a long string (length > 8)

Example

```c
#include <stdio.h>
#include "spssdio.h"
void func()
{
    int  fH;                   /* file handle        */
    int  error;                /* error code         */
    int  type;                 /* missing format type */
    int  numV;                 /* number of variables */
    int  *typesV;              /* variable types     */
    char **namesV;             /* variable names     */
    char cMiss1[9];            /* first missing value */
    char cMiss2[9];            /* second missing value*/
    char cMiss3[9];            /* third missing value */

    ...
    error = spssOpenRead("bank.sav", &fH);
    ...
    /* Print missing value information for all short string    ** variables
*/
    error = spssGetVarNames(fH, &numV, &namesV, &typesV);
    if (error == SPSS_OK)
    {
     int i;
     for (i = 0; i < numV; ++i)
     {
      if (0 < typesV[i] && typesV[i] <= 8)
      {
       /* Short string variable */
       error = spssGetVarCMissingValues
         (fH, namesV[i], &type, cMiss1, cMiss2, cMiss3);
       if (error != SPSS_OK) continue; /* Ignore error */
       printf("Variable %s, missing values: ", namesV[i]);
       switch (type)
       {
       case 0:
        printf("None\n");
        break;
       case 1:
        printf("%s\n", cMiss1);
        break;
       case 2:
        printf("%s, %s\n", cMiss1, cMiss2);
        break;
       case 3:
        printf("%s, %s, %s\n", cMiss1, cMiss2, cMiss3);
        break;
       default: /* Should never come here */
        printf("Invalid format code\n");
        break;
       }
      }
     }
     spssFreeVarNames(namesV, typesV, numV);
    }
}
```

See also **spssGetVarNMissingValues**.

spssGetVarColumnWidth

int spssGetVarColumnWidth (int *handle*, const char **varName*, int **columnWidth*)

Description

This function reports the value of the column width attribute of a variable. A value of zero is special and means that the IBM SPSS Statistics Data Editor, which is the primary user of this attribute, will set an appropriate width using its own algorithm.

Parameter	Description
handle	Handle to the data file.
varName	Variable name.
columnWidth	Pointer to column width. Non-negative.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist

spssGetVarCompatName

int spssGetVarCompatName (const int handle, const char* longName, char* shortName)

Description

When writing IBM SPSS Statistics data files, the I/O Module creates variable names which are compatible with legacy releases. These names are no more than 8 bytes in length, are all upper case, and are unique within the file. spssGetVarCompatName

allows access to these "mangled" name for input files and for output files after spssCommitHeader has been called.

Parameter	Description
handle	Handle to the data file
longName	The variable's extended name as a null-terminated string
shortName	A 9 byte character variable to receive the mangled name as a null-terminate string

Returns

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_DICT_NOTCOMMIT	spssCommitHeader has not been called for an output file
SPSS_VAR_NOTFOUND	Variable longName does not exist

spssGetVarCValueLabel

int spssGetVarCValueLabel
(int *handle*, const char *varName*, const char *value*, char *label*)

Description

This function gets the value label for a given value of a short string variable. The label is copied as a null-terminated string into the buffer *label*, whose size must be at least 61 to hold the longest possible value label (60 characters plus the null terminator). To get value labels more than 60 characters long, use the spssGetVarCValueLabelLong function.

Parameter	Description
handle	Handle to the data file
varName	Variable name
value	Short string value for which the label is wanted

label	Label for the value

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_LABELS	The variable has no labels (warning)
SPSS_NO_LABEL	There is no label for the given value (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_STR_EXP	The variable is numeric
SPSS_SHORTSTR_EXP	The variable is a long string (length > 8)
SPSS_EXC_STRVALUE	The value is longer than the length of the variable

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int  fH;                  /* file handle        */
   int  error;               /* error code         */
   char vLab[61];            /* label for the value */
   ...
   error = spssOpenRead("myfile.sav", &fH);
   ...
   /* Get and print the label for value "IL" of variable STATE */
   error = spssGetVarCValueLabel(fH, "STATE", "IL", vLab);
   if (error == SPSS_OK)
    printf("Value label for variable STATE, value \"IL\": %s\n", vLab);
   ...
}
```

spssGetVarCValueLabelLong

```
int spssGetVarCValueLabelLong
(int handle, const char *varName, const char *value, char *labelBuff,
int lenBuff, int *lenLabel)
```

Description

This function returns a null-terminated value label corresponding to one value of a specified variable whose values are short strings. The function permits the client to limit the number of bytes (including the null terminator) stored and returns the number of data bytes (excluding the null terminator) actually stored. If an error is detected, the label is returned as a null string, and the length is returned as 0.

Parameter	Description
handle	Handle to the data file
varname	Null-terminated variable name
value	Null-terminated value for which label is requested
labelBuff	Returned as null-terminated label
lenBuff	Overall size of *labelBuff* in bytes
lenLabel	Returned as bytes stored excluding terminator

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_LABELS	The variable has no labels (warning)
SPSS_NO_LABEL	The given value has no label (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist

SPSS_STR_EXP	The specified variable has numeric values
SPSS_SHORTSTR_EXP	The specified variable has long string values
SPSS_EXC_STRVALUE	The specified value is longer than the variable's data

spssGetVarCValueLabels

int spssGetVarCValueLabels
(int *handle*, const char **varName*, char ***values, char ***labels, int **numLabels*)

Description

This function gets the set of labeled values and associated labels for a short string variable. The number of values is returned as **numLabels*. Values are stored into an array of **numLabels* pointers, each pointing to a char string containing a null-terminated value, and **values* is set to point to the first element of the array. Each value string is as long as the variable. The corresponding labels are structured as an array of **numLabels* pointers, each pointing to a char string containing a null-terminated label, and **labels* is set to point to the first element of the array.

The two arrays and the value and label strings are allocated on the heap. When they are no longer needed, spssFreeVarCValueLabels should be called to free the memory.

Parameter	Description
handle	Handle to the data file
varName	Variable name
values	Pointer to array of pointers to values
labels	Pointer to array of pointers to labels
numLabels	Pointer to number of values or labels

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description

SPSS_OK	No error
SPSS_NO_LABELS	The variable has no labels (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_STR_EXP	The variable is numeric
SPSS_SHORTSTR_EXP	The variable is a long string (length > 8)
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
  int  fH;                  /* file handle              */
  int  error;               /* error code               */
  int  numL;                /* number of values or labels */
  char **cValuesL;          /* values                   */
  char **labelsL;           /* labels                   */
  ...
  error = spssOpenRead("myfile.sav", &fH);
  ...
  /* Get and print value labels for short string variable STATE */
  error = spssGetVarCValueLabels(fH, "STATE",
          &cValuesL, &labelsL, &numL);
  if (error == SPSS_OK)
  {
   int i;
   printf("Value labels for STATE\n");
   for (i = 0; i < numL; ++i)
   {
    printf("Value: %s, Label: %s\n", cValuesL[i], labelsL[i]);
   }
   /* Free the values & labels */
   spssFreeVarCValueLabels(cValuesL, labelsL, numL);
  }
}
```

See also **spssFreeVarCValueLabels**.

spssGetVarHandle

int spssGetVarHandle (int *handle*, const char **varName*, double **varHandle*)

Description

This function returns a handle for a variable, which can then be used to read or write (depending on how the file was opened) values of the variable. If *handle* is associated with an output file, the dictionary must be written with spssCommitHeader before variable handles can be obtained via spssGetVarHandle.

Parameter	Description
handle	Handle to the data file.
varName	Variable name.
varHandle	Pointer to handle for the variable. Note that the variable *handle* is a double, and not int or long.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_DICT_NOTCOMMIT	Dictionary of the output file has not yet been written with spssCommitHeader
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_NO_MEMORY	No memory available

Example

See **spssGetValueChar**.

spssGetVariableSets

int spssGetVariableSets (int *handle*, char ***varSets*)

Description

This function reports the variable sets information in the data file. Variable sets information is stored in a null-terminated string and a pointer to the string is returned in *varSets*. Since the variable sets string is allocated on the heap, the caller should free it by calling spssFreeVariableSets when it is no longer needed.

Parameter	Description
handle	Handle to the data file
varSets	Pointer to pointer to variable sets string

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_VARSETS	There is no variable sets information in the file (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "spssdio.h"
void func()
{
   int fH;                  /* file handle              */
   int error;               /* error code               */
   char *vSets;             /* ptr to variable sets info.*/
   ...
   error = spssOpenRead("bank.sav", &fH);
   ...
   /* Get & print variable sets information. */
   error = spssGetVariableSets(fH, &vSets);
   if (error == SPSS_NO_VARSETS)
   {
    printf("No variable sets information in file.\n");
   }
   else if (error == SPSS_OK)
   {
    /* In real life, we would format the variable sets
    ** information better
    */
    printf("Variable sets:\n%s", vSets);
    /* Remember to free variable set string */
    spssFreeVariableSets(vSets);
   }
   ...
}
```

See also **spssFreeVariableSets**.

spssGetVarInfo

int spssGetVarInfo (int *handle*, int *iVar*, char **varName*, int **varType*)

Description

This function gets the name and type of one of the variables present in a data file. It serves the same purpose as spssGetVarNames but returns the information one variable at a time and, therefore, can be passed to a Visual Basic program. The storage to receive the variable name must be at least 65 bytes in length because the name is returned as a null-terminated string. The type code is an integer in the range 0–32767, 0 indicating a numeric variable and a positive value indicating a string variable of that size.

Parameter	Description
handle	Handle to the data file

iVar	Zero-origin variable number
varName	Returned as the variable name
varType	Returned as the variable type

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_FILE	The data file contains no variables
SPSS_NO_MEMORY	Insufficient memory
SPSS_VAR_NOTFOUND	Parameter *iVar* is invalid

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int  fH;                  /* file handle         */
   int  error;               /* error code          */
   long count;               /* number of variables */
   int  *typeV;              /* variable type       */
   char *nameV;              /* variable name       */
   ...
   error = spssOpenRead("bank.sav", &fH);

   ...
   /* Get number of variables  */
   error = spssGetNumberofVariables(fH, &count);
   if (error == SPSS_OK)
   /* Get & print variable names and types */
   {
    int i;
    for (i = 0; i < count; ++i)
    {error = spssGetVarInfo(fH, i, nameV, typeV);
   if (error == SPSS_OK)
    printf("Variable name: %s, type: %d\n", nameV, typeV);
        }
       }
}
```

spssGetVarLabel

int spssGetVarLabel (int *handle*, const char **varName*, char **varLabel*)

Description

This function copies the label of variable *varName* into the buffer pointed to by *varLabel*. Since the variable label is at most 120 characters long and null-terminated, the size of the buffer should be at least 121. To get labels more than 120 characters long, use the spssGetVarLabelLong function.

Parameter	Description
handle	Handle to the data file
varName	Variable name
varLabel	Variable label buffer

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_LABEL	The variable does not have a label (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist

Example

```
#include <stdio.h>
 #include "spssdio.h"
 void func()
 {
    int  fH;                   /* file handle    */
    int  error;                /* error code     */
    char vLabel[121];          /* variable label */
    ...
    error = spssOpenRead("bank.sav", &fH);
    ...
    /* Get and print the label of the variable AGE */
    error = spssGetVarLabel(fH, "AGE", vLabel);
    if (error == SPSS_OK)
     printf("Variable label of AGE: %s\n", vLabel);
    ...
 }
```

spssGetVarLabelLong

int spssGetVarLabelLong (int *handle*, const char *varName*, char *labelBuff*,
int *lenBuff*, int *lenLabel*)

Description

This function returns the null-terminated label associated with the specified variable
but restricts the number of bytes (including the null terminator) returned to *lenBuff*
bytes. This length can be conveniently specified as *sizeof(labelBuff)*. The function also
returns the number of data bytes (this time excluding the null terminator) stored. If an
error is detected, the label is returned as a null string, and the length is returned as 0.

Parameter	Description
handle	Handle to the data file
varName	Null-terminated variable name
labelBuff	Buffer to receive the null-terminated label
lenBuff	Overall size of *labelBuff* in bytes
lenLabel	Returned as bytes stored excluding terminator

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_LABEL	The variable does not have a label (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist

spssGetVarMeasureLevel

int spssGetVarMeasureLevel (int *handle*, const char **varName*, int **measureLevel*)

Description

This function reports the value of the measurement level attribute of a variable.

Parameter	Description
handle	Handle to the data file.
varName	Variable name.
measureLevel	Pointer to measurement level. Set to SPSS_MLVL_NOM, SPSS_MLVL_ORD, or SPSS_MLVL_RAT, for nominal, ordinal, and scale (ratio), respectively.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid

SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist

spssGetVarNMissingValues

int spssGetVarNMissingValues
(int *handle*, const char **varName*, int **missingFormat*,
double **missingVal1*, double **missingVal2*, double **missingVal3*)

Description

This function reports the missing values of a numeric variable. The value of **missingFormat* determines the interpretation of **missingVal1*, **missingVal2*, and **missingVal3*. If **missingFormat* is SPSS_MISS_RANGE, **missingVal1* and **missingVal2* represent the upper and lower limits, respectively, of the range, and **missingVal3* is not used. If **missingFormat* is SPSS_MISS_RANGEANDVAL, **missingval1* and **missingVal2* represent the range and **missingVal3* is the discrete missing value. If **missingFormat* is neither of the above, it will be in the range 0–3, indicating the number of discrete missing values present. (The macros SPSS_NO_MISSVAL, SPSS_ONE_MISSVAL, SPSS_TWO_MISSVAL, and SPSS_THREE_MISSVAL may be used as synonyms for 0–3.)

Parameter	Description
handle	Handle to the data file
varName	Variable name
missingFormat	Pointer to missing value format code
missingVal1	Pointer to first missing value
missingVal2	Pointer to second missing value
missingVal3	Pointer to third missing value

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description

SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_NUME_EXP	The variable is not numeric

Example

```c
#include <stdio.h>
#include "spssdio.h"
void func()
{
    int  fH;                    /* file handle         */
    int  error;                 /* error code          */
    int  type;                  /* missing format type */
    int  numV;                  /* number of variables */
    int  *typesV;               /* variable types      */
    char **namesV;              /* variable names      */
    double nMiss1;              /* first missing value */
    double nMiss2;              /* second missing value*/
    double nMiss3;              /* third missing value */

    ...

    error = spssOpenRead("bank.sav", &fH);
    ...
    /*Print missing value information for all numeric variables */
    error = spssGetVarNames(fH, &numV, &namesV, &typesV);
    if (error == SPSS_OK)
    {
     int i;
     for (i = 0; i < numV; ++i)
      {
       if (typesV[i] == 0)
       {
        /* Numeric variable */
        error = spssGetVarNMissingValues
          (fH, namesV[i], &type, &nMiss1, &nMiss2, &nMiss3);
        if (error != SPSS_OK) continue; /* Ignore error */
        printf("Variable %s, missing values: ", namesV[i]);
        switch (type)
        {
        case SPSS_MISS_RANGE:
         printf("%e through %e\n", nMiss1, nMiss2);
         break;
        case SPSS_MISS_RANGEANDVAL:
         printf("%e through %e, %e\n", nMiss1, nMiss2, nMiss3);
         break;
        case 0:
         printf("None\n");
         break;
        case 1:
         printf("%e\n", nMiss1);
         break;
        case 2:
         printf("%e, %e\n", nMiss1, nMiss2);
         break;
        case 3:
         printf("%e, %e, %e\n", nMiss1, nMiss2, nMiss3);
         break;
        default: /* Should never come here */
         printf("Invalid format code\n");
         break;
        }
       }
      }
     spssFreeVarNames(namesV, typesV, numV);
    }
}
```

See also **spssGetVarCMissingValues**.

spssGetVarNValueLabel

int spssGetVarNValueLabel
(int *handle*, const char **varName*, double *value*, char **label*)

Description

This function gets the value label for a given value of a numeric variable. The label is copied as a null-terminated string into the buffer *label*, whose size must be at least 61 to hold the longest possible value label (60 characters) plus the terminator. To get value labels more than 60 characters long, use the spssGetVarNValueLabelLong function.

Parameter	Description
handle	Handle to the data file
varName	Variable name
value	Numeric value for which the label is wanted
label	Label for the value

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_LABELS	The variable has no labels (warning)
SPSS_NO_LABEL	There is no label for the given value (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_NUME_EXP	The variable is not numeric

Example

```
  #include "spssdio.h"
void func()
{
  int  fH;                    /* file handle        */
  int  error;                 /* error code         */
  char vLab[61];              /* label for the value */
  ...
  error = spssOpenRead("bank.sav", &fH);
  ...
  /* Get and print the label for value 0.0 of variable SEX */
  error = spssGetVarNValueLabel(fH, "SEX", 0.0, vLab);
  if (error == SPSS_OK)
   printf("Value label for variable SEX, value 0.0: %s\n", vLab);
  ...
}
```

spssGetVarNValueLabelLong

```
int spssGetVarNValueLabelLong
(int handle, const char *varName, double value, char *labelBuff, int lenBuff, int *lenLabel)
```

Description

This function returns a null-terminated value label corresponding to one value of a specified numeric variable. It permits the client to limit the number of bytes (including the null terminator) stored and returns the number of data bytes (excluding the null terminator) actually stored. If an error is detected, the label is returned as a null string, and the length is returned as 0.

Parameter	Description
handle	Handle to the data file
varName	Null-terminated variable name
value	Value for which label is requested
labelBuff	Returned as null-terminated label
lenBuff	Overall size of *labelBuff* in bytes
lenLabel	Returned as bytes stored excluding terminator

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_LABELS	The variable has no labels (warning)
SPSS_NO_LABEL	The given value has no label (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_NUME_EXP	The specified variable has string values

spssGetVarNValueLabels

int spssGetVarNValueLabels
(int *handle*, const char **varName*, double ***values*, char ****labels*, int **numLabels*)

Description

This function gets the set of labeled values and associated labels for a numeric variable. The number of values is returned as **numLabels*. Values are stored into an array of **numLabels* double elements, and **values* is set to point to the first element of the array. The corresponding labels are structured as an array of **numLabels* pointers, each pointing to a char string containing a null-terminated label, and **labels* is set to point to the first element of the array.

The two arrays and the label strings are allocated on the heap. When they are no longer needed, spssFreeVarNValueLabels should be called to free the memory.

Parameter	Description
handle	Handle to the data file
varName	Variable name
values	Pointer to array of double values
labels	Pointer to array of pointers to labels
numLabels	Pointer to number of values or labels

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_NO_LABELS	The variable has no labels (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_NUME_EXP	The variable is not numeric
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int  fH;                    /* file handle             */
   int  error;                 /* error code              */
   int  numL;                  /* number of values or labels */
   double *nValuesL;           /* values                  */
   char **labelsL;             /* labels                  */
   ...
   error = spssOpenRead("bank.sav", &fH);
   ...
   /* Get and print value labels for numeric variable SEX */
   error = spssGetVarNValueLabels(fH, "SEX",
           &nValuesL, &labelsL, &numL);
   if (error == SPSS_OK)
   {
    int i;
    printf("Value labels for SEX\n");
    for (i = 0; i < numL; ++i)
    {
     printf("Value: %g, Label: %s\n", valuesL[i], labelsL[i]);
    }
    /* Free the values & labels */
    spssFreeVarNValueLabels(nValuesL, labelsL, numL);
   }
}
```

See also **spssFreeVarNValueLabels**.

spssGetVarNames

int spssGetVarNames (int *handle*, int **numVars*, char ****varNames*, int ***varTypes*)

Description

This function gets the names and types of all the variables present in a data file. The number of variables is returned as **numVars*. Variable names are structured as an array of **numVars* pointers, each pointing to a char string containing a variable name, and **varNames* is set to point to the first element of the array. Variable types are stored into a corresponding array of **numVars* in elements, and **varTypes* is set to point to the first element of the array. The type code is an integer in the range 0–32767, 0 indicating a numeric variable and a positive value indicating a string variable of that size.

The two arrays and the variable name strings are allocated on the heap. When they are no longer needed, spssFreeVarNames should be called to free the memory.

Parameter	Description
handle	Handle to the data file
numVars	Pointer to number of variables
varNames	Pointer to array of pointers to variable names
varTypes	Pointer to array of variable types

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_FILE	The data file contains no variables
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
  int  fH;                  /* file handle        */
  int  error;               /* error code         */
  int  numV;                /* number of variables */
  int  *typesV;             /* variable types     */
  char **namesV;            /* variable names     */
  ...
  error = spssOpenRead("bank.sav", &fH);
  ...
  /* Get & print variable names and types */
  error = spssGetVarNames(fH, &numV, &namesV, &typesV);
  if (error == SPSS_OK)
  {
   int i;
   for (i = 0; i < numV; ++i)
   {
    printf("Variable name: %s, type: %d\n", namesV[i], typesV[i]);
   }
   /* Free the variable names & types */
   spssFreeVarNames(namesV, typesV, numV);
  }
}
```

See also **spssFreeVarNames**.

spssGetVarPrintFormat

int spssGetVarPrintFormat
(int *handle*, const char **varName*, int **printType*, int **printDec*, int **printWid*)

Description

This function reports the print format of a variable. Format type, number of decimal
places, and field width are returned as **printType*, **printDec*, and **printWid*, respectively.

Parameter	Description
handle	Handle to the data file
varName	Variable name
printType	Pointer to print format type code (file *spssdio.h* defines macros of the form SPSS_FMT_... for all valid format type codes)
printDec	Pointer to number of digits after the decimal
printWid	Pointer to print format width

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
   int  fH;                   /* file handle        */
   int  error;                /* error code         */
   int  type;                 /* print format type  */
   int  dec;                  /* digits after decimal */
   int  wid;                  /* print format width */

   error = spssOpenRead("bank.sav", &fH);
   ...
   /* Get & print the print format of variable AGE */
   error = spssGetVarPrintFormat(fH, "AGE", &type, &dec, &wid);
   if (error == SPSS_OK)
   {
    printf("Variable AGE, format code %d, width.dec %d.%d\n",
     type, wid, dec);
   }
}
```

spssGetVarRole

```
int  spssGetVarRole
(const int hFile, const char *varName, int *varRole)
```

Description

This function reports the role of a variable. The role is returned as *varRole.*

Parameter	Description

hFile	Handle to the data file
varName	Variable name
varRole	Pointer to variable role. Set to SPSS_ROLE_INPUT, SPSS_ROLE_TARGET, SPSS_ROLE_BOTH, SPSS_ROLE_NONE, SPSS_ROLE_PARTITION, or SPSS_ROLE_SPLIT.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist

spssGetVarWriteFormat

int spssGetVarWriteFormat
(int *handle*, const char *varName*, int *writeType*, int *writeDec*, int *writeWid*)

Description

This function reports the write format of a variable. Format type, number of decimal places, and field width are returned as *writeType*, *writeDec*, and *writeWid*, respectively.

Parameter	Description
handle	Handle to the data file
varName	Variable name
writeType	Pointer to write format type code (file *spssdio.h* defines macros of the form SPSS_FMT_... for all valid format type codes)
writeDec	Pointer to number of digits after the decimal
writeWid	Pointer to write format width

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist

Example

```
#include <stdio.h>
 #include "spssdio.h"
 void func()
 {
   int  fH;                 /* file handle         */
   int  error;              /* error code          */
   int  type;               /* write format type   */
   int  dec;                /* digits after decimal */
   int  wid;                /* write format width   */

   error = spssOpenRead("bank.sav", &fH);
   ...
   /* Get & print the write format of variable AGE */
   error = spssGetVarWriteFormat(fH, "AGE", &type, &dec, &wid);
   if (error == SPSS_OK)
   {
    printf("Variable AGE, format code %d, width.dec %d.%d\n",
     type, wid, dec);
   }
 }
```

spssHostSysmisVal

> void spssHostSysmisVal(double *missVal*)

Description

> This function accesses the same information as spssSysmisVal but returns the
> information via a parameter rather than on the stack as the function result. The problem
> being addressed is that not all languages return doubles from functions in the same
> fashion.

> | Parameter | Description |
> |-----------|-------------|
> | *missval* | Returned as the system missing value |

Returns

> The function always succeeds, and there is no return code.

> See also **spssSysmisVal**.

spssIsCompatibleEncoding

> int spssIsCompatibleEncoding(
>   const int hFile,
>   int* bCompatible)

Description

> This function determines whether the file's encoding is compatible with the current
> interface encoding. The result value of *bCompatible will be false when reading a code
> page file in UTF-8 mode, when reading a UTF-8 file in code page mode when reading
> a code page file encoded in other than the current locale's code page, or when reading
> a file with numbers represented in reverse bit order. If the encoding is incompatible,
> data stored in the file by other applications, particularly Data Entry for Windows, may
> be unreliable.

> | Parameter | Description |
> |-----------|-------------|

hFile	Handle to the file
bCompatible	0 for false; 1 for true

Returns

The function returns SPSS_OK or an error value:

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is invalid

spssLowHighVal

void spssLowHighVal (double *lowest*, double *highest*)

Description

This function returns the "lowest" and "highest" values used for numeric missing value ranges on the host system. It may be called at any time.

Parameter	Description
lowest	Pointer to "lowest" value
highest	Pointer to "highest" value

Returns

None

Example

```
#include "spssdio.h"
void func()
{
 int    fH;                     /* file handle    */
 int    error;                  /* error code     */
 double lowest, highest;
 ...
 error = spssOpenWrite("data.sav", &fH);
 ...
 /* Create numeric variable SALARY and set range "lowest"
 ** through 0 as missing
 */
 error = spssSetVarName(fH, "SALARY", SPSS_NUMERIC);
 if (error == SPSS_OK)
 {
  spssLowHighVal(&lowest, &highest);
  /* Last arg. is a placeholder since we are defining a range
  ** only
  */
  error = spssSetVarNMissingValues(fH, "SALARY",
   SPSS_MISS_RANGE,lowest, 0.0, 0.0);
  ...
 }
}
```

spssOpenAppend

int spssOpenAppend (const char *fileName*, int *handle*)

Description

This function opens IBM SPSS Statistics data files for appending cases and returns a
handle that should be used for subsequent operations on the file. (Note: this function
will not work correctly on compressed data files created by versions prior to release
14.0.)

There are some precautions involving encoding. If you are in UTF-8 mode, you
can't open a data file in code page. If you are in code page mode, you can't open a
system file in UTF-8. You also can't open a file in reversed bit order. If the file violates
any of these rules, spssOpenAppend returns SPSS_INCOMPATIBLE_APPEND. While in
code page mode, you can open a file in a different code page, but the results are not
predictable. For more information about encoding, see "Interface and File Encoding"
on page 2.

Parameter	Description
fileName	Name of the file

handle   Pointer to handle to be returned

Note: If you are working in code page mode but need to specify the filename in UTF-8 then use the spssOpenAppendU8 function. It is identical to the spssOpenAppend function but takes a UTF-8 encoding of the filename and converts it to the current code page. The spssOpenAppend and spssOpenAppendU8 functions are completely identical when working in UTF-8 mode.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_FITAB_FULL	File table full (too many open data files)
SPSS_FILE_OERROR	Error opening file
SPSS_NO_MEMORY	Insufficient memory
SPSS_FILE_RERROR	Error reading file
SPSS_INVALID_FILE	File is not a valid IBM SPSS Statistics data file
SPSS_NO_TYPE2	File is not a valid IBM SPSS Statistics data file (no type 2 record)
SPSS_NO_TYPE999	File is not a valid IBM SPSS Statistics data file (missing type 999 record)
SPSS_INCOMPAT_APPEND	File created on an incompatible system.

Example

```
#include "spssdio.h"
void func()
{
   int fH;                   /* file handle */
   int error;                /* error code  */
   ...
   error = spssOpenAppend("bank.sav", &fH);
   if (error == 0)
   {
    /* fH is a valid handle; process and */
    ...
    /* close file */
    error = spssCloseAppend(fH);
    ...
   }
   else
   {
    /* Handle error*/
    ...
   }
}
```

See also **spssCloseAppend**.

spssOpenRead

int spssOpenRead (const char *fileName*, int *handle*)

Description

This function opens IBM SPSS Statistics data files for reading and returns a handle that should be used for subsequent operations on the file.

Parameter	Description
fileName	Name of the file
handle	Pointer to handle to be returned

Note: If you are working in code page mode but need to specify the filename in UTF-8 then use the spssOpenReadU8 function. It is identical to the spssOpenRead function but takes a UTF-8 encoding of the filename and converts it to the current code page. The spssOpenRead and spssOpenReadU8 functions are completely identical when working in UTF-8 mode.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_FITAB_FULL	File table full (too many open data files)
SPSS_FILE_OERROR	Error opening file
SPSS_NO_MEMORY	Insufficient memory
SPSS_FILE_RERROR	Error reading file
SPSS_INVALID_FILE	File is not a valid IBM SPSS Statistics data file
SPSS_NO_TYPE2	File is not a valid IBM SPSS Statistics data file (no type 2 record)
SPSS_NO_TYPE999	File is not a valid IBM SPSS Statistics data file (missing type 999 record)

Example

```
#include "spssdio.h"
void func()
{
  int fH;                  /* file handle */
  int error;               /* error code  */
  ...
  error = spssOpenRead("bank.sav", &fH);
  if (error == 0)
  {
   /* fH is a valid handle; process and */
   ...
   /* close file */
   error = spssCloseRead(fH);
   ...
  }
  else
  {
   /* Handle error*/
   ...
  }
}
```

See also **spssCloseRead**.

spssOpenWrite

int spssOpenWrite (const char *filename*, int *handle*)

Description

This function opens a file in preparation for creating a new IBM SPSS Statistics data file and returns a handle that should be used for subsequent operations on the file.

Parameter	Description
filename	Name of the data file
handle	Pointer to handle to be returned

Note: If you are working in code page mode but need to specify the filename in UTF-8 then use the spssOpenWriteU8 function. It is identical to the spssOpenWrite function but takes a UTF-8 encoding of the filename and converts it to the current code page. The spssOpenWrite and spssOpenWriteU8 functions are completely identical when working in UTF-8 mode.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_FITAB_FULL	File table full (too many open data files)
SPSS_FILE_OERROR	Error opening file
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include "spssdio.h"
 void func()
 {
    int fH;                    /* file handle */
    int error;                 /* error code  */
    ...
    error = spssOpenWrite("dat.sav", &fH);
    if (error == 0)
    {
     /* fH is a valid handle; process and */
     ...
     /* close file */
     error = spssCloseWrite(fH);
     ...
    }
    else
    {
     /* Handle error*/
     ...
    }
 }
```

See also **spssCloseWrite**.

spssOpenWriteCopy

int spssOpenWriteCopy (const char *fileName*, const char *dictFileName*, int *handle*)

Description

This function opens a file in preparation for creating a new IBM SPSS Statistics data file and initializes its dictionary from that of an existing IBM SPSS Statistics data file. It is useful when you want to modify the dictionary or data of an existing file or replace all of its data. The typical sequence of operations is to call spssOpenWriteCopy (newFileName, oldFileName, ...) to open a new file initialized with a copy of the old

file's dictionary, then spssOpenRead (oldFileName, ...) to open the old file to access its data.

Parameter	Description
fileName	Name of the new file
dictFileName	Name of existing file
handle	Pointer to handle to be returned

Note: If you are working in code page mode but need to specify the filename in UTF-8 then use the spssOpenWriteCopyU8 function. It is identical to the spssOpenWriteCopy function but takes a UTF-8 encoding of the filename and converts it to the current code page. The spssOpenWriteCopy and spssOpenWriteCopyU8 functions are completely identical when working in UTF-8 mode.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_FITAB_FULL	File table full (too many open IBM SPSS Statistics data files)
SPSS_FILE_OERROR	Error opening new file for output
SPSS_NO_MEMORY	Insufficient memory
SPSS_FILE_RERROR	Error reading existing file
SPSS_INVALID_FILE	File is not a valid IBM SPSS Statistics data file
SPSS_NO_TYPE2	File is not a valid IBM SPSS Statistics data file (no type 2 record)
SPSS_NO_TYPE999	File is not a valid IBM SPSS Statistics data file (missing type 999 record)

spssQueryType7

int spssQueryType7(const int *handle*, const int *subType*, int **bFound*)

Description

This function can be used to determine whether a file opened for reading or append contains a specific "type 7" record. The following type 7 subtypes might be of interest:

Subtype 3. Release information

Subtype 4. Floating point constants including the system missing value

Subtype 5. Variable set definitions

Subtype 6. Date variable information

Subtype 7. Multiple response set definitions

Subtype 8. Data Entry for Windows (DEW) information

Subtype 10. TextSmart information

Subtype 11. Measurement level, column width, and alignment for each variable

Parameter	Description
handle	Handle to the data file
subtype	Specific subtype record
bFound	Returned set if the specified subtype was encountered

Returns

The result of the query is returned in parameter *bfound*—TRUE if the record subtype was encountered when reading the file's dictionary; FALSE otherwise.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_WRMODE	The file was opened for writing
SPSS_INVALID_7SUBTYPE	Parameter subtype not between 1 and MAX7SUBTYPE

spssReadCaseRecord

int spssReadCaseRecord (int *handle*)

Description

This function reads the next case from a data file into internal buffers. Values of individual variables for the case may then be obtained by calling the spssGetValueNumeric and spssGetValueChar procedures.

Parameter	Description
handle	Handle to the data file

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_FILE_END	End of the file reached; no more cases (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_WRMODE	File is open for writing, not reading
SPSS_FILE_RERROR	Error reading file

Example

See **spssGetValueChar**.

spssSeekNextCase

int spssSeekNextCase(const int *handle*, const long *caseNumber*)

Description

This function sets the file pointer of an input file so that the next data case read will be the one specified via the *caseNumber* parameter. A zero-origin scheme is used. That is, the first case is number 0. The next case can be read by calling either spssWholeCaseIn or spssReadCaseRecord. If the specified case is greater than or equal to the number of cases in the file, the call to the input function will return SPSS_FILE_END.

Parameter	Description
handle	Handle to the data file
caseNumber	Zero-origin case number

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_WRMODE	The file is open for writing, not reading
SPSS_NO_MEMORY	Insufficient memory
SPSS_FILE_RERROR	Error reading the file
SPSS_INVALID_FILE	The file is not a valid IBM SPSS Statistics data file

See also **spssWholeCaseIn**, **spssReadCaseRecord**.

spssSetCaseWeightVar

int spssSetCaseWeightVar (int *handle*, const char **varName*)

Description

This function defines variable *varName* as the case weight variable for the data file specified by the *handle*.

Parameter	Description
handle	Handle to the data file
varName	The name of the case weight variable

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_NUME_EXP	The variable is not numeric
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include "spssdio.h"
void func()
{
    int fH;                 /* file handle */
    int error;              /* error code  */
    ...
    error = spssOpenWrite("data.sav", &fH);
    ...
    /* Define variables */
    error = spssSetVarName(fH, "NUMCHILD", SPSS_NUMERIC);
    if (error == SPSS_OK)
     error = spssSetVarName(fH, "TOYPREF", SPSS_NUMERIC);
    ...
    /* Set NUMCHILD as case weight */
    error = spssSetCaseWeightVar(fH, "NUMCHILD");
    if (error != SPSS_OK)
    {
     /* Handle error */
    }
}
```

spssSetCompression

int spssSetCompression (int *handle*, int *compSwitch*)

Description

This function sets the compression attribute of a data file. Compression is set on if *compSwitch* is one and off if it is zero. If this function is not called, the output file will be uncompressed by default.

Parameter	Description
handle	Handle to the data file
compSwitch	Compression switch

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader
SPSS_INVALID_COMPSW	Invalid compression switch (other than 0 or 1)

Example

```
#include "spssdio.h"
 void func()
  {
    int fH;                 /* file handle */
    int error;              /* error code  */
    ...
    error = spssOpenWrite("data.sav", &fH);
    ...
    /* Set data compression on */
    error = spssSetCompression(fH, 1);
    ...
  }
```

spssSetDateVariables

int spssSetDateVariables (int *handle*, int *numofElements*, const long **dateInfo*)

Description

This function sets the Trends date variable information. The array at *dateInfo* is assumed to have *numofElements* elements that correspond to the data array portion of record 7, subtype 3. Its first six elements comprise the "fixed" information, followed by a sequence of one or more three-element groups. Since very little validity checking is done on the input array, this function should be used with caution and is recommended only for copying Trends information from one file to another.

Parameter	Description
handle	Handle to the data file
numofElements	Size of the array *dateInfo*
dateInfo	Array containing date variables information

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader
SPSS_INVALID_DATEINFO	The date variable information is invalid
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include <stdlib.h>
#include "spssdio.h"
void func()
{
  int fHIn, fHOut;  /* input & output file handles    */
  int error;        /* error code                     */
  long *dateInfo;   /* pointer to date variable info. */
  int nElements;    /* number of elements in date info. array */
  ...
  /* Open one file for reading and one for writing. */
  error = spssOpenRead("bank.sav", &fHIn);
  ...
  error = spssOpenWrite("bankcopy.sav", &fHOut);
  ...
  /* Get the list of variables in input file;
  ** define variables in output file
  */
  ...
  /* Get date variable information from input file and copy
  ** it to output file
  */
  error = spssGetDateVariables(fHIn, &nElements, &dateInfo);
  if (error == SPSS_OK)
  {
   error = spssSetDateVariables(fHOut, nElements, dateInfo);
   ...
   free(dateInfo);
  }
  ...
}
```

See also **spssGetDateVariables**.

spssSetDEWFirst

int spssSetDEWFirst (const int *handle*, const void **pData*, const long *nBytes*)

Description

DEW information (file information which is private to the Data Entry product) can be delivered to the I/O Module in whatever segments are convenient for the client. The spssSetDEWFirst function is called to deliver the first such segment, and subsequent segments are delivered by calling spssSetDEWNext as many times as necessary.

Parameter	Description
handle	Handle to the data file
pData	Pointer to the data to be written
nBytes	Number of bytes to write

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_EMPTY_DEW	Zero bytes to be written (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_READ_MODE	The file is not open for writing
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_NO_MEMORY	Insufficient memory for control blocks
SPSS_FILE_BADTEMP	Cannot open or write to temporary file

See also **spssSetDEWNext**.

spssSetDEWGUID

int spssSetDEWGUID (const int handle, const char* asciiGUID)

Description

This function stores the Data Entry for Windows uniqueness indicator on the data file. It should only be used by the DEW product.

Parameter	Description
handle	Handle to the data file
asciiGUID	The GUID (as a null-terminated string) to be stored on the file

Returns

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	The file is open for input or append

SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_NO_MEMORY	Insufficient memory to store the GUID

spssSetDEWNext

int spssSetDEWNext (const int *handle*, const void **pData*, const long *nBytes*)

Description

The DEW information (file information that is private to the Data Entry product) can be delivered to the I/O Module in whatever segments are convenient for the client. The spssSetDEWFirst function is called to deliver the first such segment, and subsequent segments are delivered by calling spssSetDEWNext as many times as necessary.

Parameter	Description
handle	Handle to the data file
pData	Pointer to the data to be written
nBytes	Number of bytes to write

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_DEW_NOFIRST	spssSetDEWFirst was never called
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_READ_MODE	The file is not open for writing
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_NO_MEMORY	Insufficient memory for control blocks
SPSS_FILE_BADTEMP	Cannot open or write to temporary file

See also **spssSetDEWFirst**.

spssSetFileAttributes

```
int spssSetFileAttributes(
    const int hFile,
    const char** attribNames,
    const char** attribText,
    const int nAttributes)
```

Description

This function replaces all the datafile attributes. It is the converse of
spssGetFileAttributes, and the names of subscripted attributes must contain the unit
origin subscripts in square brackets as in Prerequisite[11]. If the number of attributes is
zero, the vector pointers can be NULL, and all attributes will be discarded.

Parameter	Description
hFile	Handle to the data file
attribNames	Pointer to a vector of attribute names
attribText	Pointer to a vector of attribute values
nAttributes	The number of element in each vector

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by
positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	The file is read-only
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_INVALID_ATTRDEF	Missing name, missing text, or invalid subscript
SPSS_INVALID_ATTRNAME	Lexically invalid attribute name

spssSetIdString

int spssSetIdString (int *handle*, const char **id*)

Description

This function sets the file label of the output data file associated with *handle* to the given string *id*.

Parameter	Description
handle	Handle to the data file.
id	File label. The length of the string should not exceed 64 characters.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_EXC_LEN64	Label length exceeds 64; truncated and used (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader

Example

```
include "spssdio.h"
 void func()
{
   int fH;                    /* file handle */
   int error;                 /* error code  */
   char id[] = "This is a file label.";
   ...
   error = spssOpenWrite("data.sav", &fH);
   ...
   error = spssSetIdString(fH, id);
   if (error == SPSS_OK)
   {
    /* The label of the data file is now the string
    ** "This is a file label."
    */
    ...
   }
}
```

spssSetInterfaceEncoding

```
int spssInterfaceEncoding(
   const int iEncoding)
```

Description

Use this function to change the interface encoding. If the call is successful, all text communicated to or from the I/O Module in subsequent calls will be in the specified mode. Also, all text in files written will be in the specified mode. There can be no open files when this call is made.

Parameter	Description
iEncoding	An encoding mode, SPSS_ENCODING_CODEPAGE (the default) or SPSS_ENCODING_UTF8.

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_ENCODING	The specified encoding is not valid

SPSS_FILES_OPEN	IBM SPSS Statistics files are open

spssSetLocale

```
char* spssSetLocale(
    const int iCategory,
    const char* pszLocale)
```

Description

The I/O Module's locale is separate from that of the client application. When the I/O Module is first loaded, its locale is set to the system default. The spssSetLocale function gives the client application control over the I/O Module's locale. The parameters and return value are identical to those for the C run-time function setlocale.

Parameter	Description
iCategory	A locale category, for example LC_ALL or LC_CTYPE. These are defined in the header file *locale.h*.
pszLocale	A locale, for example "Japanese.932".

Returns

The function returns the resulting locale, for example "French_Canada.1252"

spssSetMultRespDefs

int spssSetMultRespDefs(const int *handle*, const char **mrespDefs*)

Parameter	Description
handle	Handle to the data file
mrespDefs	Code page or UTF-8 string containing definitions

Description

This function is used to write multiple response definitions to the file. The definitions are stored as a null-terminated code page or UTF-8 string based on whether the

spssGetInterfaceEncoding() type is SPSS_ENCODING_CODEPAGE or SPSS_ENCODING_UTF8.

For multiple category sets the string contains the following:
$setname=C {label length} {label} {variable list}

For multiple dichotomy sets, the string contains the following:
$setname=D{value length} {counted value} {label length} [label] {variable list}

- All multiple multiple category and multiple dichotomy sets in the data file are returned as single string, with a newline character (\n) between each set.
- All multiple response set names begin with a dollar sign and follow variable naming rules.
- For mulitple dichotomy sets, there is no space between the "D" and the integer that represents the length of the counted value.
- If there is no label for the set, the label length is 0, and there is a single blank space for the label. (So there are two blank spaces between the label length value of 0 and the first variable name.)

For example:

$mcset=C 21 Multiple Category Set mcvar1 mcvar2 mcvar3 mcvar4 \n
$mdset1=D1 1 22 Multiple Dichotomy Set mdvar1 mdvar2 mdvar3 mdvar4 \n
$mdset2=D3 Yes 0 mdvar5 mdvar6 mdvar7

Note: You cannot write "extended" multiple dichotomy sets. "Extended" multiple dichotomy sets are sets that use counted values as category labels (CATEGORYLABELS=COUNTEDVALUES in SPSS Statistics command syntax) or the variable label of the first set variable as the set label (LABELSOURCE=VARLABEL in SPSS Statistics command syntax). You can get values of extended multiple dichotomy sets with spssGetMultRespSetsDefEx, but you cannot write extended multiple dichotomy sets.

Returns

Returns one of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error

SPSS_EMPTY_MULTRESP	The string contains no definitions (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	The file is open for input or append
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_NO_MEMORY	Insufficient memory to store the definitions

spssSetTempDir

int spssSetTempDir (const char* dirName)

Description

The I/O Module spills some large object to temporary files. Normally these files reside in the directory supplied by the Windows GetTempPath function. The spssSetTempDir function permits the I/O Module client to specify a different directory.

Parameter	Description
dirName	Fully-qualified directory name as a null-terminated string

Returns

Error Code	Description
SPSS_OK	No error
SPSS_NO_MEMORY	Insufficient memory to store the path

spssSetTextInfo

int spssSetTextInfo (int *handle*, const char **textInfo*)

Description

This function sets the text data from the null-terminated string in textInfo. If the string is longer than 255 characters, only the first 255 are (quietly) used. If textInfo contains the empty string, existing text data, if any, are deleted.

Parameter	Description
handle	Handle to the data file
textInfo	Text data

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	The file is open for input or append
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_NO_MEMORY	Insufficient memory

spssSetValueChar

int spssSetValueChar (int *handle*, double *varHandle*, const char **value*)

Description

This function sets the value of a string variable for the current case. The current case is not written out to the data file until spssCommitCaseRecord is called.

Parameter	Description
handle	Handle to the data file
varHandle	Handle to the variable
value	Value of the variable as a null-terminated string. The length of the string (ignoring trailing blanks, if any) should be less than or equal to the length of the variable.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_NOTCOMMIT	Dictionary of the output file has not yet been written with spssCommitHeader
SPSS_STR_EXP	Variable associated with the handle is numeric
SPSS_EXC_STRVALUE	The value is longer than the length of the variable

Example

See **spssSetValueNumeric**.

See also **spssCommitCaseRecord**.

spssSetValueNumeric

int spssSetValueNumeric (int *handle*, double *varHandle*, double *value*)

Description

This function sets the value of a numeric variable for the current case. The current case is not written out to the data file until spssCommitCaseRecord is called.

Parameter	Description
handle	Handle to the data file
varHandle	Handle to the variable
value	Value of the variable

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_NOTCOMMIT	Dictionary of the output file has not yet been written with spssCommitHeader
SPSS_NUME_EXP	Variable associated with the handle is not numeric

Example

```
#include "spssdio.h"
 void func()
 {
    int    fH;                /* file handle      */
    int    error;             /* error code       */
    double ageH, titleH;      /* variable handles */
    double age;               /* value of AGE     */
    ...
    error = spssOpenWrite("data.sav", &fH);
    ...
    /* Create numeric variable AGE and long string variable
    ** TITLE
    */
    error = spssSetVarName(fH, "AGE", SPSS_NUMERIC);
    ...
    error = spssSetVarName(fH, TITLE, SPSS_STRING(20));
    ...
    /* Done with dictionary definition; commit dictionary */
    error = spssCommitHeader(fH);
    ...
    /* Get variable handles */
    error = spssGetVarHandle(fH, "AGE", &ageH);
    ...
    error = spssGet VarHandle(fH, "TITLE", &titleH);
    ...
    /* Construct & write cases, with AGE set to 20, 21, ... 46
    ** and TITLE set to "Super salesman"
    */
    for (age = 20.0; age <= 46.0; ++age)
    {
     error = spssSetValueNumeric(fH, ageH, age);
     ...
     error = spssSetValueChar(fH, titleH, "Super salesman")
     ...
     error = spssCommitCaseRecord(fH);
     ...
    }
    error = spssCloseWrite(fH);
    ...
 }
```

See also **spssConvertDate**, **spssConvertTime**, **spssCommitCaseRecord**.

spssSetVarAlignment

int spssSetVarAlignment (int *handle*, const char **varName*, int *alignment*)

Description

This function sets the value of the alignment attribute of a variable.

Parameter	Description
handle	Handle to the data file.
varName	Variable name.
alignment	Alignment. Must be one of SPSS_ALIGN_LEFT, SPSS_ALIGN_RIGHT, or SPSS_ALIGN_CENTER. If not a legal value, alignment is set to a type-appropriate default.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	The file is open for input or append
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist

spssSetVarAttributes

```
int spssSetVarAttributes(
   const int hFile,
   const char* varName,
   const char** attribNames,
   const char** attribText,
   const int nAttributes)
```

Description

This function is analogous to spssSetFileAttributes. It replaces all the attributes for one variable.

Parameter	Description
hFile	Handle to the data file

varName	Name of the variable
attribNames	Pointer to a vector of attribute names
attribText	Pointer to a vector of attribute values
nAttributes	The number of element in each vector

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_VAR_NOTFOUND	Named variable is not in the file
SPSS_OPEN_RDMODE	The file is read-only
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_INVALID_ATTRDEF	Missing name, missing text, or invalid subscript
SPSS_INVALID_ATTRNAME	Lexically invalid attribute name

spssSetVarCMissingValues

int spssSetVarCMissingValues
(int *handle*, const char **varName*, int *missingFormat*,
const char **missingVal1*, const char **missingVal2*, const char **missingVal3*)

Description

This function sets missing values for a short string variable. The argument *missingFormat* must be set to a value in the range 0–3 to indicate the number of missing values supplied. When fewer than three missing values are to be defined, the redundant arguments must still be present, although their values are not inspected. For example, if *missingFormat* is 2, *missingVal3* is unused. The supplied missing values must be null-terminated and not longer than the length of the variable unless the excess length is

made up of blanks, which are ignored. If the missing value is shorter than the length of the variable, trailing blanks are assumed.

Parameter	Description
handle	The handle to the data file
varName	Variable name
missingFormat	Missing format code
missingVal1	First missing value
missingVal2	Second missing value
missingVal3	Third missing value

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_STR_EXP	The variable is numeric
SPSS_SHORTSTR_EXP	The variable is a long string (length > 8)
SPSS_INVALID_MISSFOR	Invalid missing values specification (*missingFormat* is not in the range 0–3)
SPSS_EXC_STRVALUE	A missing value is longer than the length of the variable
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include <stddef.h>
#include "spssdio.h"
void func()
{
  int fH;                          /* file handle    */
  int error;                       /* error code     */
  ...
  error = spssOpenWrite("data.sav", &fH);
  ...
  /* Create short string variable TITLE and define values
  ** consisting of blanks or periods only as missing
  */
  error = spssSetVarName(fH, "TITLE", SPSS_STRING(6));
  if (error == SPSS_OK)
  {
   /* Last arg. is a placeholder since we are defining only two
   ** missing values
   */
   error = spssSetVarCMissingValues(fH, "TITLE", 2,
          "......", "      ", NULL);
   ...
  }
}
```

spssSetVarColumnWidth

int spssSetVarColumnWidth (int *handle*, const char **varName*, int *columnWidth*)

Description

This function sets the value of the column width attribute of a variable. A value of zero is special and means that the IBM SPSS Statistics Data Editor, which is the primary user of this attribute, is to set an appropriate width using its own algorithm.

Parameter	Description
handle	Handle to the data file.
varName	Variable name.
columnWidth	Column width. If negative, a value of zero is (quietly) used instead.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	The file is open for input or append
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist

spssSetVarCValueLabel

int spssSetVarCValueLabel
(int *handle*, const char **varName*, const char **value*, const char **label*)

Description

This function changes or adds a value label for the specified value of a short string variable. The label should be a null-terminated string not exceeding 60 characters in length.

Parameter	Description
handle	Handle to the data file
varName	Variable name
value	Value to be labeled
label	Label

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_INVALID_HANDLE	The file handle is not valid.
SPSS_OPEN_RDMODE	File is open for reading, not writing.
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader.
SPSS_INVALID_VARNAME	Variable name is invalid.
SPSS_VAR_NOTFOUND	A variable with the given name does not exist.
SPSS_STR_EXP	The variable is numeric.
SPSS_SHORTSTR_EXP	The variable is a long string (length > 8).
SPSS_EXC_STRVALUE	The value (*value*) is longer than the length of the variable.
SPSS_NO_MEMORY	Insufficient memory.
SPSS_INTERNAL_VLABS	Internal data structures of the I/O Module are invalid. This signals an error in the I/O Module.

Example

```
#include "spssdio.h"
 void func()
 {
    int    fH;                    /* file handle    */
    int    error;                 /* error code     */
    ...
    error = spssOpenWrite("data.sav", &fH);
    ...
    /* Create short string variable TITLE and label the value
    ** consisting of all blanks as "Did not want title"
    */
    error = spssSetVarName(fH, "TITLE", SPSS_STRING(6));
    if (error == SPSS_OK)
    {
     error = spssSetVarCValueLabel(fH, "TITLE", "      ",
        "Did not want title");
    }
 }
```

See also **spssSetVarCValueLabels**.

spssSetVarCValueLabels

int spssSetVarCValueLabels
(int *handle*, const char ***varNames, int *numVars*,
const char ***values, const char ***labels, int *numLabels*)

Description

This function defines a set of value labels for one or more short string variables. Value labels already defined for any of the given variable(s), if any, are discarded (if the labels are shared with other variables, they remain associated).

Parameter	Description
handle	Handle to the data file
varNames	Array of pointers to variable names
numVars	Number of variables
values	Array of pointers to values
labels	Array of pointers to labels
numLabels	Number of labels or values)

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error.
SPSS_INVALID_HANDLE	The file handle is not valid.
SPSS_OPEN_RDMODE	File is open for reading, not writing.
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader.
SPSS_NO_VARIABLES	Number of variables (*numVars*) is zero or negative.
SPSS_NO_LABELS	Number of labels (*numLabels*) is zero or negative.
SPSS_INVALID_VARNAME	At least one variable name is invalid.

SPSS_VAR_NOTFOUND	At least one of the variables does not exist.
SPSS_STR_EXP	At least one of the variables is numeric.
SPSS_SHORTSTR_EXP	At least one of the variables is a long string (length < 8).
SPSS_EXC_STRVALUE	At least one value is longer than the length of the variable.
SPSS_DUP_VALUE	The list of values contains duplicates.
SPSS_NO_MEMORY	Insufficient memory.
SPSS_INTERNAL_VLABS	Internal data structures of the I/O Module are invalid. This signals an error in the I/O Module.

Example

```
#include "spssdio.h"
 void func()
 {
    int    fH;                    /* file handle       */
    int    error;                 /* error code        */
    static char *vNames[2]=       /* variable names    */
     { "TITLE", "OLDTITLE" };
    static char *vValues[3] =     /* values to be labeled */
     { "       ", "techst", "consul" };
    static char *vLabels[3] =     /* corresponding labels */
     { "Unknown", "Member of tech. staff", "Outside consultant" };
    ...
    error = spssOpenWrite("data.sav", &fH);
    ...
    /* Define two short string variables TITLE & OLDTITLE and a
    ** set of shared value labels
    */
    error = spssSetVarName(fH, vNames[0], SPSS_STRING(6));
    if (error == SPSS_OK)
     error = spssSetVarName(fH, vNames[1], SPSS_STRING(6));
    if (error == SPSS_OK)
    {
     error =
      spssSetVarCValueLabels(fH, vNames, 2, vValues, vLabels, 3);
     ...
    }
 }
```

See also **spssSetVarCValueLabel**.

spssSetVarLabel

int spssSetVarLabel (int *handle*, const char *varName*, const char *varLabel*)

Description

This function sets the label of a variable.

Parameter	Description
handle	Handle to the data file.
varName	Variable name.
varLabel	Variable label. The length of the string should not exceed 120 characters. If *varLabel* is the empty string, the existing label, if any, is deleted.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_EXC_LEN120	Variable label's length exceeds 120; truncated and used (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include "spssdio.h"
void func()
{
    int  fH;                 /* file handle   */
    int  error;              /* error code    */
    ...
    error = spssOpenWrite("data.sav", &fH);
    /* Do the file operations here */
    ...
    /* Define string variable NAME of length 8 */
    error = spssSetVarName(fH, "NAME", SPSS_STRING(8));
    ...
    /* Label the variable */
    error =
     spssSetVarLabel(fH, "NAME", "Name of respondent");
    ...
}
```

spssSetVarMeasureLevel

int spssSetVarMeasureLevel (int *handle*, const char **varName*, int *measureLevel*)

Description

This function sets the value of the measurement level attribute of a variable.

Parameter	Description
handle	Handle to the data file.
varName	Variable name.
measureLevel	Measurement level. Must be one of SPSS_MLVL_NOM, SPSS_MLVL_ORD, SPSS_MLVL_RAT, or SPSS_MLVL_UNK for nominal, ordinal, scale (ratio), and unknown, respectively. If SPSS_MLVL_UNK, measurement level is set to a type-appropriate default.

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error

SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	The file is open for input or append
SPSS_DICT_COMMIT	spssCommitHeader has already been called
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_INVALID_MEASURELEVEL	measureLevel is not in the legal range, or it is SPSS_MLVL_RAT and the variable is a string variable

spssSetVarNMissingValues

int spssSetVarNMissingValues
(int *handle*, const char **varName*, int *missingFormat*,
double *missingVal1*, double *missingVal2*, double *missingVal3*)

Description

This function sets missing values for a numeric variable. The interpretation of the arguments *missingVal1*, *missingVal2*, and *missingVal3* depends on the value of *missingFormat*. If *missingFormat* is set to SPSS_MISS_RANGE, *missingVal1* and *missingVal2* are taken as the upper and lower limits, respectively, of the range, and *missingVal3* is ignored. If *missingFormat* is SPSS_MISS_RANGEANDVAL, *missingval1* and *missingVal2* are taken as limits of the range and *missingVal3* is taken as the discrete missing value. If *missingFormat* is neither of the above, it must be in the range 0–3, indicating the number of discrete missing values present. For example, if *missingFormat* is 2, *missingVal1* and *missingVal2* are taken as two discrete missing values and *missingVal3* is ignored. (The macros SPSS_NO_MISSVAL, SPSS_ONE_MISSVAL, SPSS_TWO_MISSVAL, and SPSS_THREE_MISSVAL may be used as synonyms for 0–3.)

Parameter	Description
handle	Handle to the data file
varName	Variable name
missingFormat	Missing values format code
missingVal1	First missing value

missingVal2	Second missing value
missingVal3	Third missing value

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_NUME_EXP	The variable is not numeric
SPSS_INVALID_MISSFOR	Invalid missing values specification (*missingFormat* is invalid or the lower limit of range is greater than the upper limit)
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include "spssdio.h"
void func()
{
    int    fH;                      /* file handle    */
    int    error;                   /* error code     */
    ...
    error = spssOpenWrite("data.sav", &fH);
    ...
    /* Create numeric variable BUYCODE and set range 1-9 as
    ** missing
    */
    error = spssSetVarName(fH, "BUYCODE", SPSS_NUMERIC);
    if (error == SPSS_OK)
    {
     /* Last arg. is a placeholder since we are defining a range
     ** only
     */
     error =
      spssSetVarNMissingValues(fH, "BUYCODE", SPSS_MISS_RANGE,
            1.0, 9.0, 0.0);
    ...
    }
}
```

See also **spssSetVarCMissingValues**.

spssSetVarNValueLabel

int spssSetVarNValueLabel
(int *handle*, const char *varName*, double *value*, const char *label*)

Description

This function changes or adds a value label for the specified value of a numeric variable. The label should be a null-terminated string not exceeding 60 characters in length.

Parameter	Description
handle	Handle to the data file
varName	Variable name
value	Value to be labeled
label	Label

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error.
SPSS_INVALID_HANDLE	File handle not valid.
SPSS_OPEN_RDMODE	File is open for reading, not writing.
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader.
SPSS_INVALID_VARNAME	Variable name is invalid.
SPSS_VAR_NOTFOUND	A variable with the given name does not exist.
SPSS_NUME_EXP	The variable is not numeric.
SPSS_NO_MEMORY	Insufficient memory.
SPSS_INTERNAL_VLABS	Internal data structures of the I/O Module are invalid. This signals an error in the I/O Module.

Example

```
#include "spssdio.h"
void func()
{
   int    fH;                      /* file handle   */
   int    error;                   /* error code    */
   ...
   error = spssOpenWrite("data.sav", &fH);
   ...
   /* Create numeric variable BUYCODE and label value 0.0 as
   ** "Unknown"
   */
   error = spssSetVarName(fH, "BUYCODE", SPSS_NUMERIC);
   if (error == SPSS_OK)
   {
    error =
     spssSetVarNValueLabel(fH, "BUYCODE", 0.0, "Unknown");
    ...
   }
}
```

See also **spssSetVarNValueLabels**.

spssSetVarNValueLabels

int spssSetVarNValueLabels
(int *handle*, const char **varNames, int *numVars*,
const double *values, const char **labels, int *numLabels*)

Description

This function defines a set of value labels for one or more numeric variables. Value labels already defined for any of the given variable(s), if any, are discarded (if the labels are shared with other variables, they remain associated with those variables).

Parameter	Description
handle	Handle to the data file
varNames	Array of pointers to variable names
numVars	Number of variables
values	Array of values
labels	Array of pointers to labels
numLabels	Number of labels or values

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error.
SPSS_INVALID_HANDLE	The file handle is not valid.
SPSS_OPEN_RDMODE	File is open for reading, not writing.
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader.
SPSS_NO_VARIABLES	Number of variables (*numVars*) is zero or negative.
SPSS_NO_LABELS	Number of labels (*numLabels*) is zero or negative.
SPSS_INVALID_VARNAME	At least one variable name is invalid.

SPSS_VAR_NOTFOUND	At least one of the variables does not exist.
SPSS_NUME_EXP	At least one of the variables is not numeric.
SPSS_DUP_VALUE	The list of values contains duplicates.
SPSS_NO_MEMORY	Insufficient memory.
SPSS_INTERNAL_VLABS	Internal data structures of the I/O Module are invalid. This signals an error in the I/O Module.

Example

```
#include "spssdio.h"
void func()
{
   int   fH;                   /* file handle       */
   int   error;                /* error code        */
   static char *vNames[2]=     /* variable names    */
    { "AGE", "AGECHILD" };
   static double vValues[3] =  /* values to be labeled */
    { -2.0, -1.0, 0.0 };
   static char *vLabels[3] =   /* corresponding labels */
    { "Unknown", "Not applicable", "Under 1" };
   ...
   error = spssOpenWrite("data.sav", &fH);
   ...
   /* Define two numeric variables AGE & AGECHILD and a set of
   ** shared value labels
   */
   error = spssSetVarName(fH, vNames[0], SPSS_NUMERIC);
   if (error == SPSS_OK)
    error = spssSetVarName(fH, vNames[1], SPSS_NUMERIC);
   if (error == SPSS_OK)
   {
    error =
     spssSetVarNValueLabels(fH, vNames, 2, vValues, vLabels, 3);
    ...
   }
}
```

See also **spssSetVarNValueLabel**.

spssSetVarName

int spssSetVarName (int *handle*, const char **varName*, int *varLength*)

Description

This function creates a new variable named *varName*, which will be either numeric or string based on *varLength*. If the latter is zero, a numeric variable with a default format of F8.2 will be created; if it is greater than 0 and less than or equal to 32767, a string variable with length *varLength* will be created; any other value will be rejected as invalid. For better readability, the macros SPSS_NUMERIC and SPSS_STRING(*length*) may be used as values for *varLength*.

Parameter	Description
handle	Handle to the data file
varName	Variable name
varLength	Type and size of the variable

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader
SPSS_INVALID_VARTYPE	Invalid length code (*varLength* is negative or exceeds 32767)
SPSS_INVALID_VARNAME	Variable name is invalid
SPSS_DUP_VAR	There is already a variable with the same name
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include "spssdio.h"
void func()
{
  int fH;                    /* file handle */
  int error;                 /* error code  */
  ...
  error = spssOpenWrite("data.sav", &fH);
  ...
 /* Create numeric variable AGE and string variable NAME */
  error = spssSetVarName(fH, "AGE", SPSS_NUMERIC);
  if (error == SPSS_OK)
   error = spssSetVarName(fH, "NAME", SPSS_STRING(20));
  ...
}
```

spssSetVarPrintFormat

int spssSetVarPrintFormat
(int *handle*, const char **varName*, int *printType*, int *printDec*, int *printWid*)

Description

This function sets the print format of a variable.

Parameter	Description
handle	Handle to the data file
varName	Variable name
printType	Print format type code (file *spssdio.h* defines macros of the form SPSS_FMT_... for all valid format type codes)
printDec	Number of digits after the decimal
printWid	Print format width

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error

SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_INVALID_PRFOR	The print format specification is invalid or is incompatible with the variable type
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include "spssdio.h"
void func()
{
   int fH;                  /* file handle */
   int error;               /* error code  */
   ...
   error = spssOpenWrite("data.sav", &fH);
   /* Define numeric variable TIMESTMP */
   error = spssSetVarName(fH, "TIMESTMP", SPSS_NUMERIC);
   ...
   /* Set the print format of TIMESTMP to DATETIME28.4 */
   error = spssSetVarPrintFormat(fH, "TIMESTMP",
     SPSS_FMT_DATE_TIME, 4, 28);
   ...
}
```

See also **spssSetVarWriteFormat**.

spssSetVarRole

int spssSetVarRole
(const int *hFile*, const char **varName*, const int *varRole*)

Description

This function sets the role of a variable.

Parameter	Description
hFile	Handle to the data file

varName	Variable name
varRole	Variable role. Must be one of the following values: SPSS_ROLE_INPUT, SPSS_ROLE_TARGET, SPSS_ROLE_BOTH, SPSS_ROLE_NONE, SPSS_ROLE_PARTITION, or SPSS_ROLE_SPLIT.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_INVALID_ROLE	Invalid role value

spssSetVarWriteFormat

int spssSetVarWriteFormat
(int *handle*, const char **varName*, int *writeType*, int *writeDec*, int *writeWid*)

Description

This function sets the write format of a variable.

Parameter	Description
handle	Handle to the data file
varName	Variable name
writeType	Write format type code (file *spssdio.h* defines macros of the form SPSS_FMT_... for all valid format type codes)
writeDec	Number of digits after the decimal
writeWid	Write format width

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader
SPSS_INVALID_VARNAME	The variable name is not valid
SPSS_VAR_NOTFOUND	A variable with the given name does not exist
SPSS_INVALID_WRFOR	The write format specification is invalid or is incompatible with the variable type
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include "spssdio.h"
void func()
{
   int fH;                  /* file handle */
   int error;               /* error code  */
   ...
   error = spssOpenWrite("data.sav", &fH);
   /* Define string variable ODDCHARS of length 7 */
   error = spssSetVarName(fH, "ODDCHARS", SPSS_STRING(7));
   ...
   /* Set the write format of ODDCHARS to AHEX14 */
   error =
   spssSetVarWriteFormat(fH, "ODDCHARS", SPSS_FMT_AHEX, 0, 14);
   ...
}
```

spssSetVariableSets

int spssSetVariableSets (int *handle*, const char **varSets*)

Description

This function sets the variable sets information in the data file. The information must be provided in the form of a null-terminated string. No validity checks are performed on the supplied string beyond ensuring that its length is not 0. Any existing variable sets information is discarded.

Parameter	Description
handle	Handle to the data file
varSets	Variable sets information

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_EMPTY_VARSETS	The variable sets information is empty (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_COMMIT	Dictionary has already been written with spssCommitHeader
SPSS_NO_MEMORY	Insufficient memory

Example

```
#include <stdlib.h>
#include "spssdio.h"
void func()
{
   int fHIn, fHOut;        /* input & output file handles  */
   int error;              /* error code                   */
   char *vSets;            /* ptr to variable sets info.   */
   ...
   /* Open one file for reading and one for writing. */
   error = spssOpenRead("bank.sav", &fHIn);
   ...
   error = spssOpenWrite("bankcopy.sav", &fHOut);
   ...
   /* Copy variable sets information from input file to output
   ** file
   */
   error = spssGetVariableSets(fHIn, &vSets);
   if (error == SPSS_OK)
   {
    error = spssSetVariableSets(fHOut, vSets);
    /* Handle errors and remember to free variable set string */
    ...
    free(vSets);
   }
   else if (error != SPSS_EMPTY_VARSETS)
   {
   /* Error getting variable sets information from input file */
    ...
   }
   ...
}
```

spssSysmisVal

double spssSysmisVal (void)

Description

This function returns the IBM SPSS Statistics system-missing value for the host system. It may be called at any time.

Parameter	Description
None	

Returns

The IBM SPSS Statistics system-missing value for the host system.

Example

```
#include <stdio.h>
#include "spssdio.h"
void func()
{
   double sysmis;              /* system missing value */
   ...
   /* Get and print the system missing value */
   sysmis = spssSysmisVal();
   printf("System missing value: %e\n");
   ...
}
```

spssValidateVarname

int spssValidateVarname (const char* varName)

Description

This function allows the client to validate a potential variable name. The name is checked for lexical validity only; there is no check for whether it is a duplicate name. Note that the error code SPSS_NAME_BADFIRST indicates that the name is entirely composed of valid characters but that the first character is not valid in that position, for example the name begins with a period or digit. Note also that names ending with a period are technically valid but are to be discouraged because they cause difficulty if they appear at the end of a line of syntax.

Parameter	Description
varName	null-terminated variable name

Returns

Error Code	Description
SPSS_NAME_OK	The name is valid
SPSS_NAME_SCRATCH	The name is invalid because it begins with "#"
SPSS_NAME_SYSTEM	The name is invalid because it begins with "$"
SPSS_NAME_BADLTH	The name is too long
SPSS_NAME_BADCHAR	The name contains an invalid character
SPSS_NAME_RESERVED	The name is a reserved word

SPSS_NAME_BADFIRST   The name begins with an invalid character

spssWholeCaseIn

int spssWholeCaseIn (int *handle*, char **caseRec*)

Description

This function reads a case from a data file into a case buffer provided by the user. The required size of the buffer may be obtained by calling spssGetCaseSize. This is a fairly low-level function whose use should not be mixed with calls to spssReadCaseRecord using the same file handle because both procedures read a new case from the data file.

Parameter	Description
handle	Handle to the data file
caseRec	Buffer to contain the case

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_FILE_END	End of the file reached; no more cases (warning)
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_WRMODE	File is open for writing, not reading
SPSS_FILE_RERROR	Error reading file

Example

```
#include <stdlib.h>
#include "spssdio.h"
void func()
{
   int fH;                  /* file handle      */
   int error;               /* error code       */
   int caseSize;            /* size of a case */
   char *cRec;              /* pointer to case record */
   ...
   error = spssOpenRead("bank.sav", &fH);
   ...
   /* Find out the size of the case and allocate memory for the
   ** case record.
   */
   error = spssGetCaseSize(fH, &caseSize);
   ...
   cRec = (char *) malloc(caseSize);
   ...
   error = spssWholeCaseIn(fH, cRec);
   ...
   /* Buffer cRec now contains the first case in the data file.
   ** It is up to us to make sense out of it.
   */
   ...
}
```

See also **spssGetCaseSize**, **spssWholeCaseOut**.

spssWholeCaseOut

int spssWholeCaseOut
(int *handle*, const char **caseRec*)

Description

This function writes a case assembled by the caller to a data file. The case is assumed to have been constructed correctly in the buffer *caseRec*, and its validity is not checked. This is a fairly low-level function whose use should not be mixed with calls to spssCommitCaseRecord using the same file handle because both procedures write a new case to the data file.

Parameter	Description
handle	Handle to the data file
caseRec	Case record to be written to the data file

Returns

One of the following codes. Success is indicated by zero (SPSS_OK), errors by positive values, and warnings, if any, by negative values.

Error Code	Description
SPSS_OK	No error
SPSS_INVALID_HANDLE	The file handle is not valid
SPSS_OPEN_RDMODE	File is open for reading, not writing
SPSS_DICT_NOTCOMMIT	Dictionary of the output file has not yet been written with spssCommitHeader
SPSS_FILE_WERROR	File write error

Example

```c
#include <string.h>
#include "spssdio.h"
void func()
{
   int fH;                  /* file handle   */
   int error;               /* error code    */
   int caseSize;            /* size of a case */
   char caseRec[16];        /* case record   */
   double age;              /* value of AGE  */
   ...
   error = spssOpenWrite("data.sav", &fH);
   ...
   /* Define two variables */
   error = spssSetVarName(fH, "NAME", SPSS_STRING(7));
   ...
   error = spssSetVarName(fH, "AGE", SPSS_NUMERIC);
   ...
   /* Done with dictionary definition; commit dictionary */
   error = spssCommitHeader(fH);
   ...
   /* Please note that code beyond this requires knowledge of
   ** IBM SPSS Statistics data file formats, and it very easy to produce
   ** garbage.
   */
   /* Find out the size of the case and make sure it is 16 as
   ** we assume it to be
   */
   error = spssGetCaseSize(fH, &caseSize);
   ...
   /* Construct one case with NAME "KNIEVEL" and AGE 50.
   ** Write out the case and close file.
   */
   memcpy(caseRec, "KNIEVEL ", 8); /* Padding to 8 */
   age = 50.0;
   memcpy(caseRec+8, &age, 8); /* Assuming sizeof double is 8 */
   error = spssWholeCaseOut(fH, caseRec);
   ...
   error = spssCloseWrite(fH);
   ...
}
```

See also **spssGetCaseSize**, **spssWholeCaseIn**.

Notices

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group
Attention: Licensing
233 S. Wacker Drive
Chicago, IL 60606
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, and SPSS are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at *http://www.ibm.com/legal/copytrade.shtml*.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.