VBstrAPI ver 1.0 rev 1.40 Reference Manual

CIS ID : 74131,2175
Internet : 74131.2175@compuserve.com

VBstrAPI.DLL is a Visual Basic language extension module providing medium and large arrays of strings and optimized string and file functions.

This DLL is designed for, and requires, Visual Basic for Windows.

Help File Updated: 95.06.24 - Ver 1.0 Rev 1.40

Contents

Reference

Introduction to VBstrAPI

Welcome to the VBstrAPI.DLL Visual Basic extension Library. The library was written to provide two enhancements to Visual Basic:

- An ultra-fast string search function and a smart file-copy function written *mostly in assembler*.
- Two large string handling objects: CatStr (a single 64k string manipulation object) and ArrayStr (a huge multiple string array limited in size *only* by the amount of free global memory available).

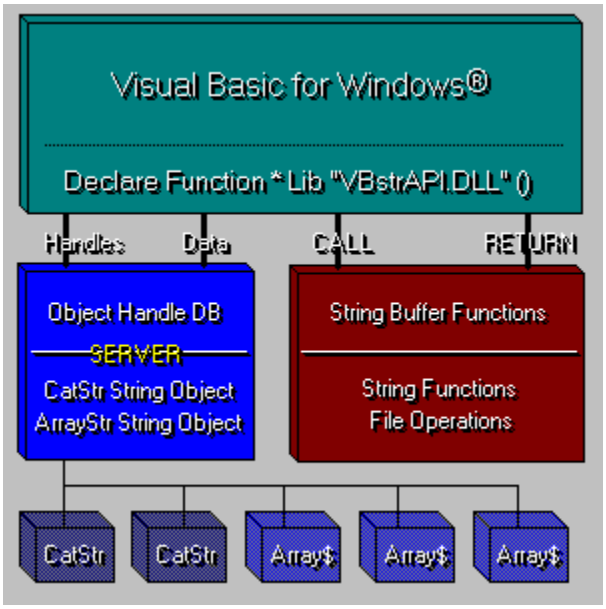
The implementation of the fast string search function (actually two: case-sensitive and case-insensitive) requires that the programmer pass a standard string to the function. This allows you to pass any string that Visual Basic handles (up to 64k).

The implementation of the two large string objects requires the DLL to maintain storage on the global heap on behalf of the application. Due to this requirement, and that many applications (or one application requiring many large strings) may need to use the library at the same time, a Client-Server paradigm was used.

Client-Server libraries must implement a method of identifying which application is accessing which object. Although there are many methods available for solving this problem, I have opted for the **unique handle** method.

When you want to create a large string object for your application you will need to call a function that will create the object and return a handle to it. Your application then uses this handle to identify the object it wants to use.

The diagram below demonstrates the relationship and differences between the two methods described.



VBstrAPI was built to support two styles of application interface: Client-Server and Procedural.

The String and File functions are procedural because they are not required to access stored data between calls.

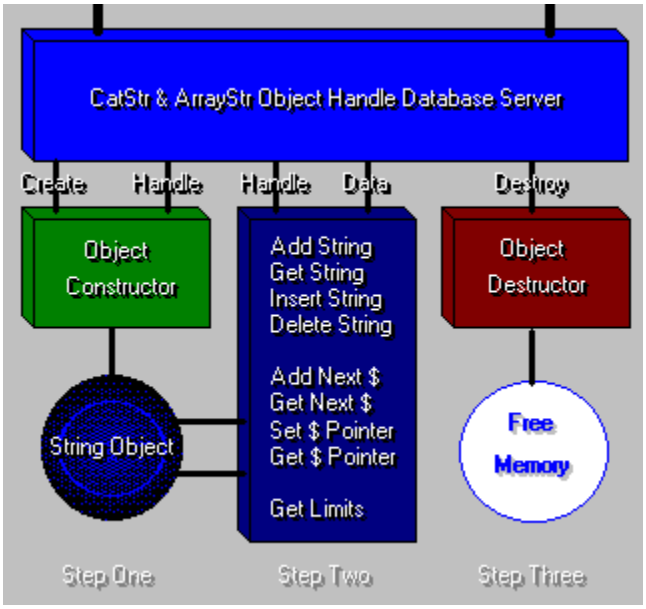
The CatStr and ArrayStr objects are Client-Server because they must store data for the calling application between method calls.

This diagram shows that functional calls return the data without maintaining storage for the calling program.

CatStr and ArrayStr objects, however, must maintain large string buffers between calls, and therefore are implemented using a Server paradigm.

The Server knows where the data is for a method call by maintaining handles to the object. For this reason, calls associated with objects require you to maintain the handle in your code. The good news is that any number of programs can use the library at once. Also, any application can have more than one instance of each object.

Proper use of the string objects requires that you understand how they are created, used and eventually destroyed. It is important that you follow the steps outlined below in the next diagram.



In this next diagram you can see the proper use and operation of CatStr and ArrayStr objects. The Object Handle Database Server returns or uses the handle to a specific object.

The first step (Step One) is to create the object. VBstrAPI returns a unique handle to the object that you will use to access it.

In the second step (Step Two) you pass the handle to the object in one of the access methods. The handle is used to insure you are accessing the correct string object.

In the third step (Step Three), which is performed after you have completed using the object, you will destroy the object, thus freeing any memory it used.

If you should forget to destroy the objects you use, their memory will be locked until the DLL is unloaded. When you are in development mode, this will usually happen automatically since Visual Basic unloads the DLL each time the program ends normally. Even if you break execution of the program abnormally.

Fortunately, using the functions provided, it is very easy to use string objects. As you will recall from the diagram above, you will use these objects in a three stage process. Notice that the two string objects use similar methods for access. The CatStr Object is accessed through fewer methods than the ArrayStr Object. The more richly defined ArrayStr Object exports a number of special purpose methods.

Create the String Object

```
' This example demonstrates the object creation
' step for CatStr objects and ArrayStr objects

Dim CHandle As Integer ' Handle to CatStr Object
Dim SHandle As Integer ' Handle to ArrayStr Object

   ' Create a CatStr Object

   CHandle = CreateNewCatString( 32768 )

   If CHandle > -1 Then

      Print "CatStr Object created! Handle is " & CHandle

   Else

      Print "Unable to create CatStr Object."

   End If

   ' Create ArrayStr Object of 10000 strings of 80 characters

   SHandle = CreateNewStringArray( 10000, 80 )

   If SHandle > -1 Then

      Print "ArrayStr Object created! Handle is " & SHandle

   Else

      Print "Unable to create String Array."

   End If
```

Use the String Object

```
' This example demonstrates sample use of
' the CatStr Object type

Dim File    As Integer ' file handle
Dim rc      As Integer ' return codes

   File = FreeFile
   Open "MyData.Fil" For Input As #File

   While Not Eof(File)

      Line Input #File, StrBuffer

      ' use CatStrAddLine to add a CR/LF terminated line

      rc = CatStrAddLine( CHandle, StrBuffer )

   Wend

   Close #File

   ' reset the Current Line Pointer

   CatStrResetCLP( CHandle )

   Print "CatStr is " & CatStrLength( CHandle )  & " characters long."

   ' use CatStrNext to scan through the string line by line

   Print "Line 1:" & CatStrNext( CHandle )
   Print "Line 2:" & CatStrNext( CHandle )
```

Destroy the String Object

```
DestroyCatString( CHandle )
DestoryStringArray( SHandle )
```

Alphabetical Reference

ArrayStrBufferSize

This function returns the declared maximum size of the string. The maximum size is defined in the **CreateNewStringArray()** method.

Visual Basic Declaration

```
Declare Function ArrayStrBufSize Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer) As Long
```

Parameters

- **SHandle As Integer**
  String Handle returned by CreateNewCatString().

Returns

- **Long**
  Length of the buffer for this string element.

Example

```
'
' Prints 81 (Which represents 80 character strings)
'
SHandle% = CreateNewStringArray( 10000,81 )
Print ArrayStrBufSize( SHandle )
```

ArrayStrClear

Used to clear the entire contents of the **ArrayStr** referenced by **SHandle.**

Visual Basic Declaration

```
Declare Sub ArrayStrClear Lib "VBstrAPI.DLL" (ByVal SHandle As Integer)
```

Parameters

- **SHandle As Integer**
  The handle to this string returned by CreateNewStringArray()

Returns

- N/A

Example

```
'
' Prints ""
'
SHandle% = CreateNewStringArray( 100,81 )

rc& = PutArrayStr( SHandle%, 0, "First String" )
rc& = PutArrayStr( SHandle%, 1, "Second String" )

ArrayStrClear SHandle%

Print GetArrayStr( SHandle%, 0 )
```

ArrayStrCLP

Returns the ***Current Line Pointer*** for the string object reference by the string handle. The CLP is used with **PutArrayNext()** and **GetArrayNext()** auto-incrementing methods.

Visual Basic Declaration

```
Declare Function ArrayStrCLP Lib "VBstrAPI.DLL" (ByVal SHandle As Integer)
As Long
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- **Long**
  The current auto-incremental line pointer.

Comments

The **CLP** is a property of the **ArrayStr** object. It is used to determine the next array element to receive a string using the **PutArrayNext** method, or retrieve a string using the **GetArrayNext** method. You can set the starting CLP by using the **ArrayStrSetCLP** method.

ArrayStrElements

Used to determine the number of elements (dimensions) defined for the string referenced by the string handle.

Visual Basic Declaration

```
Declare Function ArrayStrElements Lib "VBstrAPI.DLL" (ByVal SHandle As Integer) As Long
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- **Long**
  The number of elements assigned to this array.

Example

```
'
' Prints '128'
'
SHandle% = CreateNewStringArray( 128, 256 )
Print ArrayStrElements( SHandle% )
```

ArrayStrMemSize

Returns the actual number of bytes consumed by the **ArrayStr** object reference by the string handle.

Visual Basic Declaration

```
Declare Function ArrayStrMemSize Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer) As Long
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- **Long**
  The actual number of bytes used by this object.

Example

```
'
' Prints '20000'
'
SHandle% = CreateNewStringArray( 100, 200 )
Print ArrayStrMemSize( SHandle% )
```

ArrayStrResize

This method is used to resize the ArrayStr after it has been created. You can expand or shrink the size of the array. See PutArrayStr method for information on how ArrayStr will expand automatically.

Visual Basic Declaration

```
Declare Function ArrayStrResize Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer, ByVal NewSize As Long) As Integer
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **NewSize as Long**
  Number of elements required.

Returns

- **Integer**
  Returns 0 if successful, -1 if not.

ArrayStrSetCLP

This method is used to preset the CLP prior to using GetArrayNext and PutArrayNext methods. The CLP is the is a pointer to the next element to be stored (put) or retrieved (get).

Visual Basic Declaration

```
Declare Sub ArrayStrSetCLP Lib "VBstrAPI.DLL" (ByVal SHandle As Integer,
ByVal Element As Long)
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **Element As Long**
  The next element to use for *PutArrayNext* and *GetArrayNext* methods.

Returns

- N/A

Example

```
'
' Example of ArrayStrSetCLP method.
'
' Prints "Blue"
'
SHandle& = CreateNewStringArray( 1000, 512 )
ArrayStrSetCLP SHandle&, 500

rc& = PutArrayNext( SHandle&, "Yellow" )
rc& = PutArrayNext( SHandle&, "Blue" )

Print GetArrayStr( SHandle&, 501 )
```

CatStrAdd

Used to add (append) a new string to the **CatStr** object referenced by the string handle.

Visual Basic Declaration

```
Declare Function CatStrAdd Lib "VBstrAPI.DLL" (ByVal SHandle As Integer,
ByVal St As String) As Integer
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **St As String**
  The string to append to the current contents of the object.

Returns

- **Integer**
  Returns 0 if successful, -1 if no more room.

Example

```
'
' Prints "The moon in June is like a balloon."
'
SHandle% = CreateNewCatString( 32000 )

rc% = CatStrAdd( SHandle%, "The moon in June" )
rc% = CatStrAdd( SHandle%, " is like a balloon." )

Print CatStrCopy( SHandle% )
```

CatStrAddLine

Used to add (append) a new line to the **CatStr** object referenced by the string handle. This line will have CR/LF appended to it.

Visual Basic Declaration

```
Declare Function CatStrAddLine Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer, ByVal St As String) As Integer
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **St As String**
  The string to append to the current contents of the object. This method appends a CR/LF [Chr$(13) & Chr$(10)] to the end of the string.

Returns

- **Integer**
  Returns 0 if successful, -1 if no more room.

Example

```
'
' Prints :
' "The moon in June"
' " is like a balloon."
'
SHandle% = CreateNewCatString( 32000 )

rc% = CatStrAddLine( SHandle%, "The moon in June" )
rc% = CatStrAddLine( SHandle%, " is like a balloon." )

Print CatStrCopy( SHandle% )
```

CatStrClear

Used to clear the contents of the **CatStr** object referenced by the string handle.

Visual Basic Declaration

```
Declare Sub CatStrClear Lib "VBstrAPI.DLL" (ByVal SHandle As Integer)
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- N/A

Example

```
'
' Prints ""
'
SHandle% = CreateNewCatString( 32000 )

rc% = CatStrAdd( SHandle%, "The moon in June" )
rc% = CatStrAdd( SHandle%, " is like a balloon." )

CatStrClear SHandle%

Print CatStrCopy( SHandle% )
```

CatStrCopy

Used to copy the contents of the **CatStr** object referenced by the string handle.

Visual Basic Declaration

```
Declare Function CatStrCopy Lib "VBstrAPI.DLL" (ByVal SHandle As Integer)
As String
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- **String**
  Returns the entire contents (string) of the CatStr object.

Example

```
'
' Prints "The moon in June is like a balloon." )
'
SHandle% = CreateNewCatString( 32000 )

rc% = CatStrAdd( SHandle%, "The moon in June" )

If rc% = 0 Then

   rc% = CatStrAdd( SHandle%, " is like a balloon." )

End If

Print CatStrCopy( SHandle% )
```

CatStrFind & CatStrFindIC

This function uses the **FindString** or **FindStringIC** function described elsewhere in this manual to search for strings within *CatStr Objects*. Since CatStr Objects can be larger that normal Visual Basic strings, there is no easy way to search through them without this function. See the **FindString** function for more information.

Visual Basic Declaration

```
Declare Function CatStrFind Lib "VBstrAPI.DLL" (ByVal SHandle As Integer,
ByVal Start As Long, ByVal Target As String) As Long
```

```
Declare Function CatStrFindIC Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer, ByVal Start As Long, ByVal Target As String) As Long
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **Start As Long**
  A Long Integer is used here to allow Visual Basic to pass a number greater than 32768. Since CatStr objects can contain 65534 characters, it is necessary.
- **Target As String**
  The string you are looking for.

Returns

- **Long**
  The location in the CatStr object where the Target String was found, otherwise -1.

Example

```
'
' Note: See FindStringIC for more information.
'
' This example returns the location of "plastic" in
' the CatStr object
'
Locn& = CatStrFind( CHandle%, 1, "plastic" )
```

CatStrLength

Returns the length of the string stored by the **CatStr** object.

Visual Basic Declaration

```
Declare Function CatStrLength Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer) As Long
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- **Long**
  The length of the string stored in the object.

Example

```
'
' Prints 36
'
SHandle% = CreateNewCatString( 32000 )

rc% = CatStrAdd( SHandle%, "The moon in June" )

If rc% = 0 Then

   rc% = CatStrAdd( SHandle%, " is like a balloon." )

End If

Print CatStrLength( SHandle% )
```

CatStrLineCount

This method returns the current line count for the CatStr object refered to by SHandle.

Visual Basic Declaration

```
Declare Function CatStrLineCount Lib "VBstrAPI.DLL" ( ByVal SHandle ) As
Long
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- **LongInt**

Example

```
'
' Prints "3"
'
CHandle% = CreateNewCatString( 65535 )

rc% = CatStrAddLine( CHandle%, "This is Line 1 of Text." )
rc% = CatStrAddLine( CHandle%, "This is Line 2 of Text." )
rc% = CatStrAddLine( CHandle%, "This is Line 3 of Text." )

Print CatStrLineCount( CHandle% )

DestroyCatString CHandle%
```

CatStrLPSZ

Returns a long pointer to the zero-terminated string (*lpsz*) stored in the CatStr Object buffer.

Visual Basic Declaration

```
Declare Function CatStrLPSZ Lib "VBstrAPI.DLL" ( Byval SHandle As
Integer ) As Long
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- **Long**
  Long pointer to the string contents of the CatStr buffer.

Comments

This function is provided so the programmer can pass a pointer to the CatStr buffer to other DLLs.

CatStrResetCLP

This method resets the CLP property of the **CatStr** object. The Current Line Pointer is used internally to maintain a pointer to the next line in the objects string buffer.

Visual Basic Declaration

```
Declare Sub CatStrResetCLP Lib "VBstrAPI.DLL" (ByVal SHandle As Integer)
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- N/A

Example

```
'
' This example will print "0123456789"
'
CHandle% = CreateNewCatStr( 13 )

If CHandle% > -1 Then

   rc% = CatStrAddLine( CHandle%, "This is Line One." )
   rc% = CatStrAddLine( CHandle%, "This Is Line Two." )

   CatStrResetCLP CHandle    ' reset the CLP

   Print CatStrNextLine( CHandle%, 10 )
   DestroyCatString CHandle%

End If
```

CatStrMid$

This method is used to extract a string from a CatStr Object much as the Visual Basic Mid$ function does. This allows almost complete access to the very large string available in a CatStr Object without resorting to a complete copy.

Visual Basic Declaration

```
Declare Function CatStrMid$ Lib "VBstrAPI.DLL" (ByVal SHandle As Integer,
ByVal Start As Long, ByVal cbSize As Long)
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **Start As Long**
  Starting character of sub-string to select (1-based)
- **cbSize As Long**
  Number of characters to extract.

Returns

- **String**
  The extracted string. Returns a Null string ("") if any invalid parameters are passed. May also cause a runtime error if the handle is invalid.

CatStrNext

This **CatStr** method is used to return the next *cbSize* block of characters from the object. If the remaining number of characters is less than *cbSize* then only the remaining characters will be returned.

Visual Basic Declaration

```
Declare Function CatStrNext Lib "VBstrAPI.DLL" (ByVal SHandle As Integer,
ByVal cbSize As Long, Status As Integer) As String
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **cbSize As Long**
  Size of block to read.
- **Status As Integer**
  -1 if the method fails

Returns

- **String**
  The returned block of characters.

Example

```
'
' This example will print "0123456789"
'
CHandle% = CreateNewCatStr( 13 )

If CHandle% > -1 Then

   rc% = CatStrAdd( CHandle%, "012345" )
   rc% = CatStrAdd( CHandle%, "678910" )

   Print CatStrNext( CHandle%, 10 )
   DestroyCatString( CHandle% )

End If
```

CatStrNextLine

This method is used to return the next line from the object's string.

Visual Basic Declaration

```
Declare Function CatStrNextLine Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer, Status As Integer) As String
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **Status As Integer**
  -1 if method fails.
- **Status As Integer**
  Return code if method fails.

Returns

- **String**
  The next line. Lines are terminated with CR/LF.

Example

```
'
' This example will print "This is Line One."
'
CHandle% = CreateNewCatStr( 13 )

If CHandle% > -1 Then

   rc% = CatStrAddLine( CHandle%, "This is Line One." )
   rc% = CatStrAddLine( CHandle%, "This Is Line Two." )

   CatStrResetCLP CHandle    ' reset the CLP

   Print CatStrNextLine( CHandle%, 10 )
   DestroyCatString CHandle%

End If
```

CatStrSetCLP

This method is used to preset the CLP prior to using *CatStrNext** and *CatStrAdd** methods. The CLP is the is a pointer to the next character location within the CatStr Object buffer. You would normally want to avoid doing this, but in cases where you know the starting character of a block or line, you can use this method to set the next insert or retrieval point. It is especially useful when used with the *CatStrFind* method to move the CLP to the text located by the search.

Visual Basic Declaration

```
Declare Sub CatStrSetCLP Lib "VBstrAPI.DLL" (ByVal CHandle As Integer,
ByVal NewCLP As Long)
```

Parameters

- **CHandle As Integer**
  The reference handle for this string object.
- **NewCLP As Long**
  The new character index.

Returns

- N/A

CenterString

This general function centers the string in a string *cbSize* characters in length. If *cbSize* is less than or equal to the length of *srcStr* then *srcStr* is returned unchanged.

Visual Basic Declaration

```
Declare Function CenterString Lib "VBstrAPI.DLL" (ByVal srcStr As String,
ByVal cbSize As Integer) As String
```

Parameters

- **srcStr As String**
  The string to be centered.
- **cbSize As Integer**
  The field length to center the string.

Returns

- **String**
  The modified string.

Example

```
'
' prints "          HELLO          "
'
Centered$ = CenterString( "HELLO", 15 )
Print Centered$
```

CenterStringIn

This general function centers the string in a string *cbSize* characters in length by padding the string with the character *Char.* If *cbSize* is less than or equal to the length of *srcStr* then *srcStr* is returned unchanged.

Visual Basic Declaration

```
Declare Function CenterStringIn Lib "VBstrAPI.DLL" (ByVal srcStr As
String, ByVal Char As String, ByVal cbSize As Integer) As String
```

Parameters

- **srcStr As String**
  The string to be centered.
- **Char As String**
  The character used to fill the left and right sides of the string.
- **cbSize As Integer**
  The field length to center the string.

Returns

- **String**
  The modified string.

Example

```
'
' prints "**********HELLO**********"
'
Centered$ = CenterStringIn( "HELLO", "*", 15 )
Print Centered$
```

CopyFile

This general function copies a source file to a destination file using a very fast assembly language routine. The source and destination files are checked before the copy operation begins.

Constants **Select this button to view return codes for this function.**

Visual Basic Declaration

```
Declare Function CopyFile Lib "VBstrAPI.DLL" (ByVal srcFile As String,
ByVal destFile As String) As Integer
```

Parameters

- **srcFile As String**
  Filename of the source file (copy from).
- **destFile As String**
  Filename of the destination file (copy to).

Returns

- **Integer**
  Success if 0, otherwise an error

Example

```
'
' An example of the CopyFile() function
'
rc% = CopyFile("C:\Windows\Win.INI", "D:\BackUp\Win.INI" )

If rc% < 0 Then Print "Copy unsuccessful"
```

Constants

VBstrAPI.DLL Return Codes Reference

CopyFile Return Codes

```
'   CopyFile Error Codes

Global Const CF_SUCCESS = 0              ' Successful
Global Const CF_SAME_ERROR = -1          ' Can't copy to same file
Global Const CF_ATTR_ERROR_SRC = -2      ' source file attribute error
Global Const CF_ATTR_ERROR_DEST = -3     ' destination file attribute
error
Global Const CF_OPEN_ERROR_SRC = -4      ' source file open error
Global Const CF_OPEN_ERROR_DEST = -5     ' destination file open error
Global Const CF_READ_ERROR = -6          ' source file read error
Global Const CF_WRITE_ERROR = -7         ' destination file write error
Global Const CF_CLOSE_ERROR = -8         ' destiation file close error
Global Const CF_ATTR_ERROR = -9          ' general attribute error
Global Const CF_FILE_INVALID = -10       ' source file is invalid
Global Const CF_PATH_INVALID = -11       ' destination file is invalid
```

CreateNewCatString

Used to create a new **CatStr** object. CatStr Objects provide methods for accessing a single (up to) 65535 character string. VBstrAPI.DLL is capable of handling up to 4,096 CatStr Objects. That represents 268,435,456 bytes of memory.

Visual Basic Declaration

```
Declare Function CreateNewCatString Lib "VBstrAPI.DLL" (ByVal cbSize As
Long) As Integer
```

Parameters

- **cbSize As Long**
  The maximum size of the string buffer for the object. Values from 2 to 65536 are valid. If a number greater that 64k is used, the string length is set to 64k.

Returns

- **Long**
  Returns an SHandle (string handle) to the newly created CatStr object.

Example

```
'
' Prints "The moon in June is like a balloon." )
'
SHandle% = CreateNewCatString( 32000 )

rc% = CatStrAdd( SHandle%, "The moon in June" )
rc% = CatStrAdd( SHandle%, " is like a balloon."

Print CatStrCopy( SHandle% )
```

CreateNewStringArray

Used to create a new *ArrayStr* object.

The absolute maximum number of objects available to **VBstrAPI.DLL** is **32,767** objects.
The absolute number of items in a *ArrayStr* object is **2,147,483,646** items. The maximum size of a string element is **65,535** characters (65,536 bytes.)

Visual Basic Declaration

```
Declare Function CreateNewStringArray Lib "VBstrAPI.DLL" (ByVal cbItems As Long, ByVal cbSize As Long) As Integer
```

Parameters

- **cbItems As Long**
  Number of items (or elements) to create. This defines the dimension of the ArrayStr. Maximum is determined by the amount of available memory up to a maximum of 2,147,483,646 items.

  For example: 4 MBytes free (4,194,304 Bytes) will support 16,384 strings of 255 characters (cbSize would be 256).
- **cbSize As Long**
  The size of each item (or element) string. Maximum is 64k. Remember that cbSize is set to the maximum string length you want ***plus 1***!

  Also, if you assign a size greater than 65536, the object will default to 65536.

Returns

- **Integer**
  Returns the SHandle that identifies this object.

  If there is not enough memory to allocate the object, this method returns -1.

Example

```
'
' An example of how to create, use and destroy an ArrayStr Object
'

' Create a string array using 16,777,216 bytes of memory
' That is 16,384 strings of 1023 characters (1024 bytes)
'
' You should note that Windows can take quite awhile to
' move and allocate that much memory.

Print "Creating Working Buffer"

MousePointer = 11 ' hourglass

SHandle% = CreateNewStringArray( 16384, 1024 )

MousePointer = 0 ' normal

Print "Buffer Created."

File% = FreeFile

Open "Large.Txt" For Input As #File%

Print "Loading File"

MousePointer = 11

While Not Eof(File%)

   Line Input #File%, Buffer$
   rc% = PutArrayNext( SHandle%, Buffer$ )

   If rc% < 0 Then ' something went wrong.

WEnd

Close #File%

MousePointer = 0

Print "File Loaded."

LineCount& = ArrayStrCLP( SHandle% )

Print "Loaded " & LineCount& & " lines."

' Show first 4 lines

For ii& = 0 To 3

   Print "Line #" & ii& & ":" & GetArrayStr(SHandle%, ii&)

Next

DestroyStringArray SHandle%
```

End

DeleteArrayStr

Method used to delete an element from the array. This moves all other elements above this one down to fill in the space.

Visual Basic Declaration

```
Declare Function DeleteArrayStr Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer, ByVal Element As Long) As Integer
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **Element As Long**
  The element to delete. ArrayStr objects are zero-based arrays. The first element is zero (0).

Returns

- **Integer**
  Returns 0 if successful, otherwise -1.

Example

```
'
' This example prints "Line Three"
'
SHandle% = CreateNewStringArray( 3, 11 )

rc% = PutArrayStr( SHandle%, 0, "Line One" )
rc% = PutArrayStr( SHandle%, 1, "Line Two" )
rc% = PutArrayStr( SHandle%, 2, "Line Three" )

' Delete line two (element #1)

rc% = DeleteArrayStr( SHandle%, 1 )

' Print the current contents of element #1

Print GetArrayStr( SHandle%, 1 )

DestroyStringArray SHandle%
```

DestroyCatString

This method is used to reclaim the memory used by the **CatStr** object referenced by *SHandle*.

Visual Basic Declaration

```
Declare Sub DestroyCatString Lib "VBstrAPI.DLL" (ByVal SHandle As Integer)
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- N/A

Comments

It is important to remember to destroy all CatStr Objects before your application exits. If you do not, VBstrAPI.DLL will retain the memory allocated until it is removed. As the DLL removes itself from Windows, it will clean up any memory not handled by your application.

When in development mode, remember to remove the DLL after any UAE or other crashes that your program might cause. This will reclaim any memory retained by the DLL.

DestroyStringArray

This method is used to reclaim the memory used by the **ArrayStr** object referenced by *SHandle*.

Visual Basic Declaration

```
Declare Sub DestroyStringArray Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer)
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- N/A

Comments

It is important to remember to destroy all ArrayStr Objects before your application exits. If you do not, VBstrAPI.DLL will retain the memory allocated until it is removed. As the DLL removes itself from Windows, it will clean up any memory not handled by your application.

When in development mode, remember to remove the DLL after any UAE or other crashes that your program might cause. This will reclaim any memory retained by the DLL.

FindString & FindStringIC

These two general functions can search a string up to 65535 characters long for a string. **FindString** is case-sensitive while **FindStringIC** is *case-insensitive.*

Both functions use a powerful search procedure **written in assembler language**. There is a noticeable difference in the speed of the **FindString** functions and the comparable Visual Basic **InStr** function.

Visual Basic Declaration

```
Declare Function FindString Lib "VBstrAPI.DLL" (ByVal start As Integer,
ByVal srcStr As String, ByVal targetStr As String) As Long
Declare Function FindStringIC Lib "VBstrAPI.DLL" (ByVal start As Integer,
ByVal srcStr As String, ByVal targetStr As String) As Long
```

Parameters

- **start As Integer**
  Starting location in the string. The character at which to begin the search.
- **srcStr As String**
  The string you are searching.
- **targetStr As String**
  The string you are looking for.

Returns

- **Long**
  The location in the srcStr where the targetStr was found, otherwise -1.

Example

```
'
' An example of FindStringIC
'
    File% = FreeFile

    SearchStr$ = "Potato"

    ' open and read the file to search

    Buffer$ = Space$(16384)
    Open "SearchMe.Txt" For Binary Access Read As #File%
    Get #File%, , Buffer$
    Close #File%

    MousePointer = 11

    ' search for string ignoring case

    idx& = FindStringIC(1, Buffer$, SearchStr$ )

    If idx& > 0 Then

        Count% = 0
        ii& = 1

        While idx& > 0

            ' search for string ignoring case

            idx& = FindStringIC(ii&, Buffer$, SearchStr$)

            If idx& > 0 Then

                Count% = Count% + 1 ' count this one
                ii& = idx& + 1      ' set pointer for next search

            End If

        Wend

        Print SearchStr$ " found " & Count% & " times."

    Else

        Print SearchStr$ & " not found."

    End If

    MousePointer = 0
```

PutArrayBlk & GetArrayBlk

These specialized methods are provided to allow the programmer to store any data type into a ArrayStr Object element. These methods are far more vulnerable to errors on the part of the programmer than any of the others!

These methods are capable of storing and retrieving any non-variant data type including user-defined types (structures). When using user-defined structure types, take care when determining the length of the structure. Some programmers use variable structure definitions and this method may not work in all cases.

Visual Basic Declaration

```
Declare Sub GetArrayBlk Lib "VBstrAPI.DLL" (ByVal SHandle As Integer,
ByVal Element As Long, Block As Any, ByVal cbSize As Long)

Declare Function PutArrayBlk Lib "VBstrAPI.DLL" (ByVal SHandle As Integer,
ByVal Element As Long, Block As Any, ByVal cbSize As Long) As Integer
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **Element As Long**
  The string element index.
- **Block**
  Any variable to store in the element selected.
- **cbSize as Long**
  Size of the item to store (Max: 65535)

PutArrayBlk Returns

- **Integer**
  Returns 0 if successful, -1 if not.

Comments

Extreme care should be taken when using these methods. The cbSize parameter should indicate the exact size of the variable to store. Do not pass a length greater than the size of the variable as this will, in all likelihood, result in a UAE or other system error.

Example

```
'
' Example usage of PutArrayBlk
'
' Prints "Louis Armstrong"
'
Type SampleRecord

   Name As String * 60
   Age  As Integer
   Sex  As String * 1

End Type

Dim MyRecord As SampleRecord

MyRecord.Name = "Louis Armstrong"
MyRecord.Age = 35
MyRecord.Sex = "M"

SHandle% = CreateNewStringArray( 2, Len(MyRecord) )

rc% = PutArrayBlk( SHandle, 0, MyRecord, Len(MyRecord) )

MyRecord.Name = ""
MyRecord.Age = 0
MyRecord.Sex = ""

GetArrayBlk SHandle, 0, MyRecord, Len(MyRecord)

Print MyRecord.Name
```

GetArrayNext

This method is used to return the next string element from the **ArrayStr** object's array. When the ***Next*** methods are used to put and get strings, the CLP property is updated to index the next element. You can read and alter the CLP using the **ArrayStrCLP** and **ArrayStrSetCLP** methods.

Visual Basic Declaration

```
Declare Function GetArrayNext Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer) As String
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.

Returns

- **String**
  The contents of the string element indicated by the internal **CLP** property.

Example

```
'
' An example of GetArrayNext
'
SHandle% = CreateNewStringArray( 10, 40 )
If SHandle% > -1 Then

   ' auto-store strings into the ArrayStr

   rc& = PutArrayNext( SHandle%, "Line 0" )
   rc& = PutArrayNext( SHandle%, "Line 1" )
   rc& = PutArrayNext( SHandle%, "Line 2" )
   rc& = PutArrayNext( SHandle%, "Line 3" )
   rc& = PutArrayNext( SHandle%, "Line 4" )

   ArrayStrSetCLP(0)                    ' reset the CLP property to 0

   For ii& = 0 To 4

      Print GetArrayNext( SHandle% )     ' get the next string element

   Next

   DestroyStringArray SHandle%         ' release the array's memory

End If
```

GetArrayStr

This method is used to get a specified string from the string array. In contrast to the **GetArrayNext** method, this method doe **not** alter the **CLP** property.

Visual Basic Declaration

```
Declare Function GetArrayStr Lib "VBstrAPI.DLL" (ByVal SHandle As Integer,
ByVal Element As Long) As String
```

- **SHandle As Integer**
  The reference handle for this string object.
- **Element As Long**
  The zero-based element number.

Returns

- **String**
  The contents of the string element.

Example

```
'
' An example of GetArrayStr
'
SHandle% = CreateNewStringArray( 5, 7 )
If SHandle% > -1 Then

   ' store strings into the ArrayStr

   rc& = PutArrayStr( SHandle%, 0, "Line 0" )
   rc& = PutArrayStr( SHandle%, 1, "Line 1" )
   rc& = PutArrayStr( SHandle%, 2, "Line 2" )
   rc& = PutArrayStr( SHandle%, 3, "Line 3" )
   rc& = PutArrayStr( SHandle%, 4, "Line 4" )

   For ii& = 0 To 4

      Print GetArrayStr( SHandle%, ii& )     ' get the indexed string
element

   Next

   DestroyStringArray SHandle%         ' release the array's memory

End If
```

InsertArrayStr

The ArrayStr method inserts a string into the array at the point indicated by the element number. All string elements past this element are moved forward. The contents of the last element, if used, is lost.

Visual Basic Declaration

```
Declare Function InsertArrayStr Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer, ByVal Element As Long, ByVal St As String) As Integer
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **Element As Long**
  The element to insert at. ArrayStr objects are zero-based arrays. The first element is zero (0).
- **St As String**
  String to insert at the supplied element number.

Returns

- **Integer**
  Returns 0 if successful, otherwise -1.

Example

```
'
' This example prints
'   "Inserted Line"
'   "Line Two"
'
SHandle% = CreateNewStringArray( 4, 11 )

rc% = PutArrayStr( SHandle%, 0, "Line One" )
rc% = PutArrayStr( SHandle%, 1, "Line Two" )
rc% = PutArrayStr( SHandle%, 2, "Line Three" )

' Insert at line two (element #1)

rc% = InsertArrayStr( SHandle%, 1, "Inserted Line." )

' Print the current contents of element #1

Print GetArrayStr( SHandle%, 1 )
Print GetArrayStr( SHandle%, 2 )

DestroyStringArray SHandle%
```

PutArrayNext

This method is used to store a string into the next string element in the **ArrayStr** object's array. When the ***Next*** methods are used to put and get strings, the CLP property is updated to index the next element. You can read and alter the CLP using the **ArrayStrCLP** and **ArrayStrSetCLP** methods.

> ***Begining with Version 1.0 Revision 1.40, this function now grows the Array object if necessary and possible. Now you can create an array of ONE element and let PutArrayNext grow the array as necessary.***

Visual Basic Declaration

```
Declare Function PutArrayNext Lib "VBstrAPI.DLL" (ByVal SHandle As
Integer, ByVal St As String) As Integer
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **St As String**
  The string to be stored in the next element.

Returns

- **Integer**
  Returns 0 if successful, -1 if not.

Example

```
'
' An example of PutArrayNext
'
SHandle% = CreateNewStringArray( 1, 40 )
If SHandle% > -1 Then

   ' auto-store strings into the ArrayStr

   rc& = PutArrayNext( SHandle%, "Line 0" )
   rc& = PutArrayNext( SHandle%, "Line 1" )
   rc& = PutArrayNext( SHandle%, "Line 2" )
   rc& = PutArrayNext( SHandle%, "Line 3" )
   rc& = PutArrayNext( SHandle%, "Line 4" )

   ArrayStrSetCLP(0)                    ' reset the CLP property to 0

   For ii& = 0 To ArrayStrElements(SHandle%) - 1

      Print GetArrayNext( SHandle% )    ' get the next string element

   Next

   DestroyStringArray SHandle%         ' release the array's memory

End If
```

PutArrayStr

This method is used to store a string into a string element in the **ArrayStr** object's array.

Visual Basic Declaration

```
Declare Function PutArrayStr Lib "VBstrAPI.DLL" (ByVal SHandle As Integer,
ByVal Element As Long, ByVal St As String) As Integer
```

Parameters

- **SHandle As Integer**
  The reference handle for this string object.
- **Element As Long**
  The string element index.
- **St As String**
  The string to be stored in the next element.

Returns

- **Integer**
  Returns 0 if successful, -1 if not.

Example

```
'
' An example of PutArrayStr
'
SHandle% = CreateNewStringArray( 10, 40 )
If SHandle% > -1 Then

   ' store strings into the ArrayStr

   rc& = PutArrayStr( SHandle%, 0, "Line 0" )
   rc& = PutArrayStr( SHandle%, 1, "Line 1" )
   rc& = PutArrayStr( SHandle%, 2, "Line 2" )
   rc& = PutArrayStr( SHandle%, 3, "Line 3" )
   rc& = PutArrayStr( SHandle%, 4, "Line 4" )

   For ii& = 0 To 4

      Print GetArrayStr( SHandle%, ii& )     ' get the next string element

   Next

   DestroyStringArray SHandle%         ' release the array's memory

End If
```

Functional Reference

VBstrAPI.DLL Functional Reference by Category

Functional Reference

VBstrAPI.DLL Functional Reference by Category

Functional Reference

VBstrAPI.DLL Functional Reference by Category

Functional Reference

VBstrAPI.DLL Functional Reference by Category

Functional Reference

VBstrAPI.DLL Functional Reference by Category

Limitations

Limitations

- CatStr String Buffer Object
- Handles a maximum of 4096 global objects.
- Maximum string length is 65533 (65534 bytes)
- ArrayStr Huge String Array Object

Limitations

- CatStr String Buffer Object
- ArrayStr Huge String Array Object
- Total number of objects limited by available memory.
- Maximum number of string array elements limited only by memory.
- Maximum string array element size is 65,534 bytes.

Registration

To register VBstrAPI.DLL, you must obtain a registered version from the author. The registered version can then be included with your program. This will disable the shareware registration dialog that appears whenever the DLL is loaded or used by your program. You will also receive the latest version of the library. You will also receive a registration key that will work on all subsequent bug-fix and minor revision releases until a new version is released.

Obtaining a Registered Version of the DLL

To obtain a registered version you must send the registration amount to:

```
Greg Truesdell
Suite 308
633 North Road
Coquitlam, BC
CANADA
V3J 1P3

Or CompuServe SWREG# 4760
```

Registration Fee Options:

- CompuServe SWREG ID# 4760: US$19.95

  The registered version will be sent to you via CompuServe E-Mail within 24 hours of receipt. You will also receive a ZIP archive containing the distribution files.

- Mail: US$19.95 + US$3.50 S&H

  The registered version will be sent to you via return mail. You will also receive a 3½ disk containing the distribution files. The package will be mailed to you within 24 hours after receiving your payment. With this option you MUST send a MONEY ORDER made out to GREG TRUESDELL.

Registration Form:

Note: All registration information is held in the strictest of confidence.

VBstrAPI Function Library (DLL) v1.0 Registration Form

Mail this registration form to:

Greg Truesdell
Suite 308
633 North Road
Coquitlam, BC
CANADA
V3J 1P3

or E-Mail to:

CIS User ID :74131,2175
Internet :74131.2175@compuserve.com

Registered User Name: [
]

Company Name: [
]

Address: [
]
[
]
[
]

City: [ ]
State/Prov/etc: [ ]
Zip/Postal Code: [ ]

Tel: [ ]

E-Mail or CIS User ID: [ ]

Fee Option: [] E-Mail (US$19.95+US$2.00 S&H) [] Regular Mail (US$19.95+US$3.50 S&H)

REMEMBER: PAYMENT MUST BE MADE BY MONEY ORDER made payable to GREG TRUESDELL.
Checks will not be accepted unless you are a resident of BRITISH COLUMBIA, CANADA.

History of Changes

History of Changes

History of Changes

- Ver 1.0 Rev 1.00 (Release 1.10) 95.02.18
- Ver 1.0 Rev 1.20 (Release 1.12)
- Corrected return codes from CopyFile
- Minor internal code changes
- Ver 1.0 Rev 1.21
- Ver 1.0 Rev 1.22
- Ver 1.0 Rev 1.30
- Ver 1.0 Rev 1.31
- Ver 1.0 Rev 1.32
- Ver 1.0 Rev 1.33
- Ver 1.0 Rev 1.40 (95.06.24)

History of Changes

History of Changes

- Ver 1.0 Rev 1.00 (Release 1.10) 95.02.18
- Ver 1.0 Rev 1.20 (Release 1.12)
- Ver 1.0 Rev 1.21
- Ver 1.0 Rev 1.22
- Added the CatStrLPSZ Method
- Ver 1.0 Rev 1.30
- Ver 1.0 Rev 1.31
- Ver 1.0 Rev 1.32
- Ver 1.0 Rev 1.33
- Ver 1.0 Rev 1.40 (95.06.24)

History of Changes

History of Changes

History of Changes

- Ver 1.0 Rev 1.00 (Release 1.10) 95.02.18
- Ver 1.0 Rev 1.20 (Release 1.12)
- Ver 1.0 Rev 1.21
- Ver 1.0 Rev 1.22
- Ver 1.0 Rev 1.30
- Ver 1.0 Rev 1.31
- Ver 1.0 Rev 1.32
- Added CatStrMid$ Method
- Added CatStrFindIC and changed CatStrFind Method behavior
- Corrected FindString and FindStringIC speed problem (Thanks, Jim Moran!)
- Ver 1.0 Rev 1.33
- Ver 1.0 Rev 1.40 (95.06.24)

History of Changes

- Ver 1.0 Rev 1.00 (Release 1.10) 95.02.18
- Ver 1.0 Rev 1.20 (Release 1.12)
- Ver 1.0 Rev 1.21
- Ver 1.0 Rev 1.22
- Ver 1.0 Rev 1.30
- Ver 1.0 Rev 1.31
- Ver 1.0 Rev 1.32
- Ver 1.0 Rev 1.33
- Corrected problem with CatStrMid$ if size of segment selected was equal to the size of the string. (Thank's Colin Younger!)
- Ver 1.0 Rev 1.40 (95.06.24)

History of Changes

- Ver 1.0 Rev 1.00 (Release 1.10) 95.02.18
- Ver 1.0 Rev 1.20 (Release 1.12)
- Ver 1.0 Rev 1.21
- Ver 1.0 Rev 1.22
- Ver 1.0 Rev 1.30
- Ver 1.0 Rev 1.31
- Ver 1.0 Rev 1.32
- Ver 1.0 Rev 1.33
- Ver 1.0 Rev 1.40 (95.06.24)
- Modified the behavior of PutArrayStr to grow the array when the upper limit is reached.
- Added the ArrayStrResize method to dynamically change the size of the array.

Copyright

Your use of VBstrAPI.DLL indicates your acceptance of the following terms and conditions:

VBstrAPI.DLL ("the Library") is a Windows/Visual Basic DLL licensed by Greg L. Truesdell ("GLT").

Shareware license.

You are free to distribute the entire unmodified contents of the distribution package to anyone you wish. You may NOT distribute any other programs that utilizes the Library without obtaining a Registered User License for the Library from GLT. For a period of no more than 30 days, you may use, test and duplicate the enclosed version of the Library. Thereafter if you wish to continue using the Library you must register the Library with GLT, or else you must cease all use of the Library. **You will be an infringer if you do not pay the registration fee and continue to use this version of the Library for more than 30 days.**

Registered User License.

If you pay the registration fee for the Library to GLT, GLT will grant a non-exclusive development license for one natural person to use one copy of the software regardless if the owner of the license is a person or a business ("the Licensee"). In addition the Licensee may distribute the VBstrAPI.DLL ("the DLL") with any or all products that use the DLL with the exceptions that (a) the recipients of any such program ("the Recipients") are not licensed to use the DLL or the Library except with the products produced by Licensees, and (b) the Recipients may not further redistribute the DLL, and (c) the product using the DLL cannot enable the user to produce other programs using the DLL or other parts of the supplied distribution package. No purported transfer of the license shall be effective until the licensee notifies GLT of the name and address of the person receiving the license ("the Transferee"), and transfers all copies of the Library to the Transferee, and removes or destroys any other copies of the Library in the possession of, or under the control of the Licensee.

Disclaimer of Warranties.

GLT makes no claims as to the suitability of the Library for any specific purpose. GLT DISCLAIMS ANY AND ALL WARRANTIES EXPRESS OR IMPLIED, WRITTEN OR ORAL, INCLUDING ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY SPECIFIC PURPOSE. The 30 day evaluation period is considered liberal enough for you to determine the fitness of this product to your application.

Limitation of Liability.

In no event shall GLT be liable for any damages whatsoever arising out of the use of the Library, including without limitation any direct, indirect or consequential damages or any damages for business interruption, loss of profits, loss of information, or any pecuniary

loss even if GLT has been notified of the possibility of such damages. The limitation or exclusion of liability for incidental or consequential damages may not be allowed in some states, and in these states those particular prohibited limitations do not apply.

Copyright Information

Glossary

A

ArrayStr

C

CatStr

CLP

H

Handle

S

SHandle

Index

V

VBstrAPI Reference Manual

Title

ArrayStr

Huge String Array Object. Capable of storing, accessing and stepping through an unlimited number of (up to) 64k strings.

CatStr

Concatentation String Object. Capable of storing, accessing and line by line stepping a 64k string.

CLP

CLP = Current Line Pointer
An internal property of CatStr and ArrayStr objects which determines the next line to be read or written using ...AddLine and ...NextLine methods.

Handle

A 16 bit integer representing an object. VBstrAPI returns a handle when an object is created, and requires the handle whenever the object is referenced.

SHandle

String Handle

An integer value returned by CreateNewStringArray() and CreateNewCatString() methods. This handle is used to reference this specific string object.

A value of -1 is returned by Create* methods if no more objects are available or if the size of the string defined will not fit in available memory.