AA-Array.DLL Version 1.0
Extended Arrays for Visual Basic

AA-Array.DLL manages the creation, accessing, saving to and restoring from disk (persistence) of extended arrays. The size of these extended arrays is only limited by the amount of virtual memory on your computer. Extended arrays have one column, hence are lists of elements.

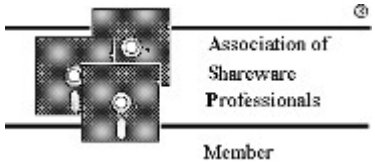
Extended arrays do **not** have the normal Visual Basic (<32,767 elements and <64,000 bytes of string space) limits. Extended arrays can have any number of elements up to 4 Gigs and use any amount of string space. Data types of an extended array element can be: Integer, Long, Single, Double, Currency, or String. In the registered version, array elements can be of User defined types.

AA-Array optimizes access to and storage of sparse arrays (i.e. arrays that have many possible elements, but few element have anything in them). For persistent arrays, AA-Array reads and writes extended arrays extremely quickly with just one function call.

AA-Array.DLL is distributed as Shareware. Try before you buy. If you continue using it, you are expected to register.

If there is functionality that you need, let us know. We will consider adding it to a future version of the DLL. Also have a look at our Custom and Semi-Custom DLL offer if you have very specific needs.

AA-Software International is a member of the Association of Shareware Professionals (ASP). See the ASP's Ombudsman Statement to know how to protect your rights.



AA-Software International
12 ter Domaine Du Bois Joli
06330 Roquefort-Les-Pins, France
Tel: (+33) 93.77.50.47
Fax: (+33) 93.77.19.78
Internet: cswilly@acm.org
CompuServe: 100343,2570

AA-Array.DLL Version 1.0 Documentation

Quick Start Example

```
'Make sure the file AA-Array.DLL is in your DOS Path
'Put the lines below in the declaration section of a form
Declare Function AryOpenString Lib "AA-Array.dll" (
      ByVal aryName_s As String, ByVal mode As Integer,
      ByVal password_s As String) As Integer
Declare Function AryClose Lib "AA-Array.dll" (
      ByVal ary_h As Integer) As Integer
Declare Sub ArySetBounds Lib "AA-Array.dll" (
      ByVal ary_h As Integer, ByVal minElement As Long,
      ByVal maxElement As Long)
Declare Sub AryGetBounds Lib "AA-Array.dll" (
      ByVal ary_h As Integer, minElement As Long, maxElement As Long)
Declare Sub ArySetString Lib "AA-Array.dll" (
      ByVal ary_h As Integer, ByVal row_l As Long, value As String)
Declare Sub AryGetString Lib "AA-Array.dll" (
      ByVal ary_h As Integer, ByVal row_l As Long, value As String)


' Put the following in the startup for a form

'Create extended string array
Dim strAry_h As Integer
strAry_h = AryOpenString("c:\tstStr.ary",
      AryCreateNew + AryPersistent, "")
If strAry_h < 0 Then
   Me.Print "Cannot create extended array."
   Me.Print "Error code is: "; strAry_h
   Exit Sub
End If

'Set the array's lower and upper bounds
'Bounds can be from -2,147,483,648 to +2,147,483,647
Const firstElement = 2147483547
ArySetBounds strAry_h, firstElement, firstElement + 100

'Set the odd elements.
'Even are not set and do not use extra memory
Dim dummyString As String
dummyString = Space(1311)
Dim i As Long
For i = firstElement To firstElement + 100 Step 2
   ArySetString strAry_h, i, Str$(i) & dummyString
Next i

'Close the extended array and free up memory
Dim retval As Integer
retval = AryClose(strAry_h)
If retval < 0 Then
   Me.Print "Cannot write extended array to disk."
   Me.Print "Error code is: "; strAry_h
   Exit Sub
End If

'Open extended string array
strAry_h = AryOpenString("c:\tstStr.ary",
```

```
      AryUseExisting + AryPersistent, "")
If strAry_h < 0 Then
   Me.Print "Cannot open extended array."
   Me.Print "Error code is: "; strAry_h
   Exit Sub
End If

'Get the array's lower and upper bounds
Dim lowerBound As Long
Dim upperBound As Long
AryGetBounds strAry_h, lowerBound, upperBound
Me.Print lowerBound, upperBound

'Get the every element (even elements will be null strings)
Dim retString As String
For i = lowerBound To upperBound
   AryGetString strAry_h, i, retString
   Me.Print i, Trim$(retString)
Next i

'Close extended array and free up memory
retval = AryClose(strAry_h)
If retval < 0 Then
   Me.Print "Cannot write extended array to disk."
   Me.Print "Error code is: "; strAry_h
   Exit Sub
End If
```

Benefits of Registering

- Additional routines for User defined types: AryOpen, ArySetElement, and AryGetElement.
- Additional *production* DLL; smaller, faster, and has the error trapping messages used by the Visual Basic programmer for debugging removed. This will allow you to distribute a truely professional version of your Visual Basic program.
- License to distribute the *production* DLL as part of your Visual Basic program.
- No Shareware *nag* screen on start-up and exit.
- Easily printable documentation in MS-Write and MS-Word V2 format.
- Technical support via FAX, CompuServe, or Internet mail for one year.
- You will be notified when new versions are available.
- You will have special reduced prices for new versions.
- You will receive the source code and a license to distribute a modified version of the production DLL as part of your Visual Basic program.

How to Register

For your convenience, a file ORDER.TXT is included in this distribution. Just edit it and print it off.

PAYMENT OPTIONS:

Registration can be made in U.S. dollars, Pounds Sterling, or in French Francs. Please check the payment option and your choice of shipping method:

U.S. Dollars

|_|	$30.00	Registration Fee
|_|	$5.00	Handling and shipping by air mail
|_|	$10.00	Handling and shipping by CompuServe mail

Pounds Sterling

|_|	£19.00	Registration Fee
|_|	£3.00	Handling and shipping by air mail
|_|	£6.00	Handling and shipping by CompuServe mail

French Francs

|_|	150.00F	Registration Fee
|_|	30.00F	Handling and shipping by air mail
|_|	60.00F	Handling and shipping by CompuServe mail

PAYMENT METHODS:

CASH or CHECK:

At this time we can accept cash or checks drawn in one of the above currencies. Please make your checks payable to: C. Scott Willy.

SWREG:

You may use CompuServe's Shareware Registration Database to register. Please use SWREG Registration ID: 4112.

For shipping by Air Mail, choose "Shipping & Handling Fees Region" U.S.A ($5.00).. For shipping by CompuServe Mail, choose "Shipping & Handling Fees Region": Central/South America ($10.00). The total amount, registration fee and shipping & handling fees will be charged to your CompuServe account.

SEND TO:

C. Scott Willy
AA-Software International
12 ter Domaine Du Bois Joli
06330 Roquefort-Les-Pins, France
Tel: (+33) 93.77.50.47
Fax: (+33) 93.77.19.78
Internet: cswilly@acm.org
CompuServe: 100343,2570
X400: (C=US; A=CompuServe; P=csmail; D=ID:100343,2570)

Installing the DLL

There are three versions of the AA-Array.DLL. One comes with the Shareware distribution. The development and production versions come when you register. Make sure the version you want to use is on the DOS path and has the name AA-ARRAY.DLL.

Shareware Version

Copy AA-ARRAY.DLL to a directory on the DOS path, in the Windows directory or in the System directory.

Registered Version

Make sure the Shareware version is no longer in a directory on the DOS path, in the Windows directory or in the System directory.

Copy AA-ARRAY.DEV to a directory on the DOS path, in the Windows directory or in the System directory and rename it AA-ARRAY.DLL.

For Distribution

If you are registered, you may distribute the production version of the DLL, AA-ARRAY.PRO, with your Visual Basic program.

Make sure that neither the Shareware version or the development version of the DLL are in a directory on the DOS path, in the Windows directory or in the System directory.

Copy AA-ARRAY.PRO to a directory on the DOS path, in the Windows directory or in the System directory and rename it AA-ARRAY.DLL.

AA-Array.DLL API Description

The API for AA-Array.DLL is a set of easy to use subroutines and functions. To use it, as with all DLL's, there are two basic steps:

1. Tell Visual Basic about the subroutines and functions by using a Declare statement.
2. Make the actual call.

All of the subroutines and functions in AA-Array.DLL are declared for you in the file `AA-ARRAY.TXT`. Cut and paste the routines that you need into the Declaration section of your Visual Basic program. More information on calling procedures in DLL's is in Chapter 24 of the *Programmer's Guide for Visual Basic*.

The method for using an extended array is similar to file I/O in that the array must be opened and closed before and after use. The basic steps are:

1. Create a new or open an existing extended array with a call to a AryOpen* function.
2. Determine or set the array's bounds with calls to AryGetBounds and ArySetBounds functions.
3. Set and get elements of the array with calls to ArySet* and AryGet* element subroutines.
4. End the use of the array and/or save out any changes to the array with AryAbort and AryClose functions.

Errors are reported by most routines by changing the extended array status. The status can be examined by calling AryGetStatus. AryOpen* and AryClose* report errors in the returned values.

Note: The '*' is a place holder for the rest of the function name. The routines AryOpen*, ArySet*, and ArgGet* are groups of related routines for simple Visual Basic types, and User Defined Types.

Create or Open an Extended Array

AryOpen is used to create an extended array or read one from a persistent file. It must be called before any access to the array. It returns a handle to the array which you pass to other routines.

There are two versions of AryOpen: one for simple Visual Basic types, and one for User Defined Types. See Using User Defined Types for more information on using AryOpen (registered version only).

```
AryOpen* ( aryName_s, mode_n, password_s )
```

Opens an extended array named `arrayName_s`. If successful returns a handle >= 0, otherwise returns an error code less than zero indicating the problem. The '`*`' is replaced by: `Integer`, `Long`, `Single`, `Double`, `Currency`, or `String`.

If the array is persistent (exists between invocations), then `arrayName_s` defines the file name to use. It must be a valid MS-DOS filename. If `password_s` is not a null string, it is used to encrypt and decrypt the array information when saving and restoring from disk. If `password_s` is not a null string, then the persistent file is encrypted. The method used is a simple exclusive-or algorithm and is only intended to keep honest people honest.

`mode_n` tells how extended array will be created or opened. For connivance, the following constants are provided in the file `AA-ARRAY.TXT`.

```
AryUseExisting      AryCreateNew
AryReadWrite        AryReadOnly
AryPersistent       AryNonPersistent
```

These constants should be added together to get the mix of modes needed. For example to open a new persistent extended array, for read/write use:

```
mode_n = AryCreateNew  + AryReadWrite + AryPersistent
```

Errors Returned

Errors are reported in the returned value. Valid array handles are greater or equal to zero. Errors are less than zero.

- AryErrNoFreeSlots = -1
- AryErrPersistentFileNotFound = -2
- AryErrMemAllowExtendedArray = -5
- ArrErrInvalidUserType = -8
- ArrErrReadPersistentFile = -11
- ArrErrParsingUserType = -12

AryOpen* Declarations

AryOpen* Constant Definitions

AryOpen* Mode Bit Definitions

AryOpen* Declarations

```
'Open
Declare Function AryOpen Lib "AA-Array.dll" (ByVal aryName_s As String, ByVal
      userTypeDefinition_s As String, ByVal mode As Integer, ByVal password_s
      As String) As Integer
Declare Function AryOpenInteger Lib "AA-Array.dll" (ByVal aryName_s As String,
      ByVal mode As Integer, ByVal password_s As String) As Integer
Declare Function AryOpenLong Lib "AA-Array.dll" (ByVal aryName_s As String,
      ByVal mode As Integer, ByVal password_s As String) As Integer
Declare Function AryOpenSingle Lib "AA-Array.dll" (ByVal aryName_s As String,
      ByVal mode As Integer, ByVal password_s As String) As Integer
Declare Function AryOpenDouble Lib "AA-Array.dll" (ByVal aryName_s As String,
      ByVal mode As Integer, ByVal password_s As String) As Integer
Declare Function AryOpenCurrency Lib "AA-Array.dll" (ByVal aryName_s As
      String, ByVal mode As Integer, ByVal password_s As String) As Integer
Declare Function AryOpenString Lib "AA-Array.dll" (ByVal aryName_s As String,
      ByVal mode As Integer, ByVal password_s As String) As Integer
```

AryOpen* Constant Definitions

```
Const AryUseExisting = 0
Const AryCreateNew = 1
Const AryReadWrite = 0
Const AryReadOnly = 2
Const AryPersistent = 0
Const AryNonPersistent = 4
```

AryOpen* Mode Bit Definitions

Bit	Zero	One
0	Use an existing array.	Create new array (erases the contents any previously created persistent arrays).
1	Constant array (read-only). Array must already exist.	Variable array (read/write).
2	Non-persistent array. Array elements are initialized to be empty. Array is not written to disk on close.	Persistent array. If mode-bit 0 = 0 then array is loaded from file. Array is written to disk on close.

Managing Array Bounds `(ArySetBounds, AryGetBounds, AryCheckIndex)`

Extended array bounds should be set and checked before use. Bounds will automatically be grown as needed, but this is not always the most efficient method of use.

`ArySetBounds ary_h, minElement_l, maxElement_l`

Sets the minimum and maximum elements for an extended array.

This function can be used to: 1) optimize creation of an array to a known size and 2) truncate a previously existing array.

Note: Calls to `ArySetElement` will grow the bounds for the extended array automatically if needed. But by setting the bounds to the maximum expected size, access will be faster and memory will be better used.

Errors Reported

Errors can be checked after the call of this routine by calling AryGetStatus.
AryErrInvalidHandle = -3
AryErrMemAllowExtendedArray = -5

`AryGetBounds_n ary_h, minElement_l, maxElement_l`

After an extended array is opened, `AryGetBounds` retrieves the current bounds for the array. This is useful when using a persistent file to know how big is the stored array.

Errors Reported

Errors can be checked after the call of this routine by calling AryGetStatus.
AryErrInvalidHandle = -3

`AryCheckIndex ary_h, index_l`

Returns TRUE if `index_l` is within current bounds.

Errors Reported

Errors can be checked after the call of this routine by calling AryGetStatus.
AryErrInvalidHandle = -3

ArySetBounds, AryGetBounds, and AryCheckIndex Declarations

ArySetBounds, AryGetBounds, and AryCheckIndex Declarations

```
'Bounds
Declare Sub ArySetBounds Lib "AA-Array.dll" (ByVal ary_h As Integer, ByVal
      minElement As Long, ByVal maxElement As Long)
Declare Sub AryGetBounds Lib "AA-Array.dll" (ByVal ary_h As Integer,
      minElement As Long, maxElement As Long)
Declare Function AryCheckIndex Lib "AA-Array.dll" (ByVal ary_h As Integer,
      ByVal row_l As Long) As Integer
```

Accessing Elements

`ArySet*` and `AryGet*` routines set and retrieve the values of individual elements of the extended array.

There are two versions of ArySet* and AryGet*: one for simple Visual Basic types, and one for User Defined Types. See Using User Defined Types for more information on using user defined types (registered version only).

```
ArySet* ary_h, row_l, value

ArySetElement ary_h, row_l, value 'registered version only
```

The '`*`' is replaced by: `Integer`, `Long`, `Single`, `Double`, `Currency`, or `String`. The data type of `value` matches the name of the routine. For example: for the routine `ArySetLong` the data type of `value` it `Long`.

Sets the entry for the specified by `row_l` to value. See Using User Defined Types for more information on using ArySetElement.

If `row_l` is outside the extended array's current bounds, the array is automatically grown to the size of `row_l`. See ArySetBounds for performance implications.

Errors Reported

Errors can be checked after the call of this routine by calling AryGetStatus.
AryErrInvalidHandle = -3
AryErrMemAllowExtendedArray = -5

```
AryGet* ary_h, row_l, value

AryGetElement ary_h, row_l, value 'registered version only
```

The '`*`' is replaced by: `Integer`, `Long`, `Single`, `Double`, `Currency`, or `String`. The data type of `value` matches the name of the routine. For example: for the routine `AryGetLong` the data type of `value` it `Long`.

Retrieves the value for the entry for the specified by `row_l`.

If `row_l` is outside the extended array's current bounds, the array is automatically grown to the size of `row_l`. See ArySetBounds for performance implications.

Errors Reported

Errors can be checked after the call of this routine by calling AryGetStatus.
AryErrInvalidHandle = -3

ArySet* and AryGet* Declarations

ArySet* and AryGet* Declarations

```
'Set Elements
Declare Sub ArySetElement Lib "AA-Array.dll" (ByVal ary_h As Integer, ByVal
      row_l As Long, value As Any)
Declare Sub ArySetInteger Lib "AA-Array.dll" Alias "ArySetElement" (ByVal
      ary_h As Integer, ByVal row_l As Long, value As Integer)
Declare Sub ArySetLong Lib "AA-Array.dll" Alias "ArySetElement" (ByVal ary_h
      As Integer, ByVal row_l As Long, value As Long)
Declare Sub ArySetSingle Lib "AA-Array.dll" Alias "ArySetElement" (ByVal ary_h
      As Integer, ByVal row_l As Long, value As Single)
Declare Sub ArySetDouble Lib "AA-Array.dll" Alias "ArySetElement" (ByVal ary_h
      As Integer, ByVal row_l As Long, value As Double)
Declare Sub ArySetCurrency Lib "AA-Array.dll" Alias "ArySetElement" (ByVal
      ary_h As Integer, ByVal row_l As Long, value As Currency)
Declare Sub ArySetString Lib "AA-Array.dll" (ByVal ary_h As Integer, ByVal
      row_l As Long, value As String)

'Get Elements
Declare Sub AryGetElement Lib "AA-Array.dll" (ByVal ary_h As Integer, ByVal
      row_l As Long, value As Any)
Declare Sub AryGetInteger Lib "AA-Array.dll" Alias "AryGetElement" (ByVal
      ary_h As Integer, ByVal row_l As Long, value As Integer)
Declare Sub AryGetLong Lib "AA-Array.dll" Alias "AryGetElement" (ByVal ary_h
      As Integer, ByVal row_l As Long, value As Long)
Declare Sub AryGetSingle Lib "AA-Array.dll" Alias "AryGetElement" (ByVal ary_h
      As Integer, ByVal row_l As Long, value As Single)
Declare Sub AryGetDouble Lib "AA-Array.dll" Alias "AryGetElement" (ByVal ary_h
      As Integer, ByVal row_l As Long, value As Double)
Declare Sub AryGetCurrency Lib "AA-Array.dll" Alias "AryGetElement" (ByVal
      ary_h As Integer, ByVal row_l As Long, value As Currency)
Declare Sub AryGetString Lib "AA-Array.dll" (ByVal ary_h As Integer, ByVal
      row_l As Long, value As String)
```

Ending Use of an Array (AryClose and AryAbort)

It is important to properly end your use of an extended array. Otherwise you risk a loss of data and windows memory leaks.

`AryClose ( ary_h )`

Frees memory resources. If the extended array was open as a persistent file, `AryClose` writes to file and flushes buffers. If must be called to ensure that persistent arrays are written to disk.

Errors Returned

Errors are reported in the returned value. Valid array handles are greater or equal to zero. Errors are less than zero.

AryErrInvalidHandle = -3

ArrErrCreatePersistentFile = -6

`AryAbort ary_h`

Frees memory resources. Does **NOT** write extended array to file no matter how it was opened.

Errors Reported

None.

AryClose and AryAbort Declarations

AryClose and AryAbort Declarations

```
'Close
Declare Function AryClose Lib "AA-Array.dll" (ByVal ary_h As Integer) As
      Integer
Declare Sub AryAbort Lib "AA-Array.dll" (ByVal ary_h As Integer)
```

Deleting Elements

Deleting elements can be an efficient way of managing your program's use of memory.

`AryDeleteAll ary_h`

Deletes all elements in a extended array. Frees memory resources. Array handle is still valid. Upper and lower bounds are unchanged.

Errors Reported

Errors can be checked after the call of this routine by calling AryGetStatus.
AryErrInvalidHandle = -3

`AryDeleteElement ary_h, row_l`

Deletes element specified by `row_l` in a extended array. Frees memory resources. Upper and lower bounds are unchanged.

Errors Reported

Errors can be checked after the call of this routine by calling AryGetStatus.
AryErrInvalidHandle = -3

AryDeleteAll and AryDeleteElement Declarations

AryDeleteAll and AryDeleteElement Declarations

```
'Deletes
Declare Sub AryDeleteElement Lib "AA-Array.dll" (ByVal ary_h As Integer, ByVal
      row_l As Long)
Declare Sub AryDeleteAll Lib "AA-Array.dll" (ByVal ary_h As Integer)
```

Getting Information (AryIsEmptyElement, AryGetStatus, and AryVersion)

These routines provide information about elements of an extended array, the status of the array, and the version of the AA-Array.DLL.

`AryIsEmptyElement ( ary_h, row_l )`

Returns TRUE if element specified by `row_l` contains all zeros and there are no strings or fixed strings allocated in this element.

Errors Reported

Errors can be checked after the call of this routine by calling AryGetStatus.
AryErrInvalidHandle = -3

`AryGetStatus (l ary_h As Integer) As Integer`

Returns the status of an extended array. An error code is negative. Error codes are reported by the previous call of a routine using extended array handle, `ary_h`.

Error Returned

AryErrInvalidHandle = -3
Previously reported error code

`AryVersion ( info_i )`

The DLL provides a routine, `AryVersion`, that returns a Visual Basic string with information about the DLL.

The following are valid values for `info_i` and what is returned

Value	Description of What is Returned
0	Version information formatted ready to display
1	Program Name
2	Version Number
3	Version Date
4	Type of DLL: Test or Production version

Errors Reported

None

AryIsEmptyElement, AryGetStatus and AryVersion Declarations

AryIsEmptyElement, AryGetStatus and AryVersion Declarations

```
'Information
Declare Function AryIsEmptyElement Lib "AA-Array.dll" (ByVal ary_h As Integer,
      ByVal row_l As Long) As Integer
Declare Function AryGetStatus Lib "AA-Array.dll" (ByVal ary_h As Integer) As
      Integer
Declare Function AryVersion Lib "AA-Array.dll" (ByVal info_i As Integer) As
      String
```

Using User Defined Types (registered version only)

The registered version of the DLL contains three routines for working with extended arrays of User Defined Types. The routines are: `AryOpen`, `ArySetElement`, and `AryGetElement`.

It is imperative that the string you pass, `userTypeDefinition_s`, *when calling AryOpen accurately describes the user defined type. If it does not, the* `ArySetElement`*, and* `AryGetElement` *will not work correctly at best and at worst will create a GPF (general protection fault!).*

```
AryOpen ( aryName_s, userTypeDefinition_s, mode_n, password_s )
```

AryOpen is in the registered version only. The parameters for opening an extended array of User Defined Types are the same as for simple Visual Basic types except it has one extra parameter. `userTypeDefinition_s` describes the User Defined Type. All other parameters have the same meaning. See Create or Open an Extended Array for information on the other parameters.

To describe a User defined type, you use normal Visual Basic data type names and separate them by a comma. Valid type names are: Integer, Long, Single, Double, Currency, and String. Use the normal Visual Basic convention of String*n to define a Fixed String of n characters.

Below is an example:

```
'Declare the User Defined Type
Type TEST
   s  As String
   fs As String * 5
   l  As Long
   i  As Integer
   c  As Currency
End Type

'Define a constant that describes User Defined Type TEST
'Must be all on one line
Global Const TestTypeDeclaration_s = "String, String*5, Long, Integer,
      Currency"

'Define an array of TEST elements
OneElement As TEST
dim ary_h as integer
ary_h = AryOpen ( "test.ary", TestTypeDeclaration_s, 0, "" )
ArySetElement ary_h, 1, OneElement
AryGetElement ary_h, 1, OneElement
```

Error Code List

All errors are reported with values less than zero. They are reported either as returned values (AryOpen* and AryClose*) or can be retrieved with calls to AryGetStatus.

AryErrNoFreeSlots = -1
AryErrPersistentFileNotFound = -2
AryErrInvalidHandle = -3
AryErrFileHeaderSize = -4
AryErrMemAllowExtendedArray = -5
ArrErrCreatePersistentFile = -6
ArrErrBoundsError = -7
ArrErrInvalidUserType = -8
ArrErrElementNotAllowcated = -9
ArrErrUserTypesMisMatch = -10
ArrErrReadPersistentFile = -11
ArrErrParsingUserType = -12

AryErrNoFreeSlots = -1
A maximum of 20 extended arrays can be opened at the same time.

AryErrPersistentFileNotFound = -2
An extended array opened in Read-only and persistent modes, must have an existing persistent file. This error is returned if the file cannot be found or is unreadable.

AryErrInvalidHandle = -3
The array handle is not valid.

AryErrFileHeaderSize = -4
The persistent file is corrupted.

AryErrMemAllowExtendedArray = -5
Memory cannot be allocated. This is usually because your system has run out of virtual memory.

ArrErrCreatePersistentFile = -6
Cannot create persistent file. This can be due to an invalid file name, no free disk space, etc.

ArrErrBoundsError = -7
Not reported.

ArrErrInvalidUserType = -8
When parsing the user type definition (`userTypeDefinition_s`) passed to AryOpen, an unrecognized type was given. This is typically a typo on your part. Valid user types are: Integer, Long, Single, Double, Currency, and String.

ArrErrElementNotAllowcated = -9
Not reported.

ArrErrUserTypesMisMatch = -10
Types specified in AryOpen do not match types previously saved in the persistent file.

ArrErrReadPersistentFile = -11
Error reading the persistent file.

ArrErrParsingUserType = -12
Cannot parse the description of the user defined type. This is generally do to a typo in the type description.

Technical Support

All registered owners of AA-Array.DLL have one year of technical support available via Fax, electronic mail or postal mail.

AA-Software International is always looking for ways to improve its products and product lines. Please document each item clearly and provide printed examples, if possible. Please be sure to include this form with all requests. It will help to make sure we can provide you with the best possible service. Please include the information requested in the file, `PROBLEM.TXT`, Problem or Enhancement Suggestion Report.

Contacting Technical Support

Internet
cswilly@acm.org

CompuServe
100343,2570

X400
(C=US; A=CompuServe; P=csmail; D=ID:100343,2570)

Fax
(+33) 93.77.19.78

Postal Mail
AA-Software International
12 ter Domaine Du Bois Joli
06330 Roquefort-Les-Pins, France

Custom and Semi-Custom DLL's

Custom DLL's are a good way to keep to the Visual Basic spirit and at the same time give your product a competitive advantage. AA-Software International will quote custom DLL's.

Doing custom work of any kind can be expensive. At the same time, a DLL that has the exact functionality you need is a big advantage keeping your software small and fast. This is why AA-Software International designs all of its DLL products to be able to be combined in a semi-custom DLL. For a fixed fee, based on the number of routines and products needed, we will create your very own DLL.

See `CUSTOM.TXT` to see how easy it is to construct your own DLL.

Legal Matters

Definition of Shareware

Shareware distribution gives users a chance to try software before buying it. If you try a Shareware program and continue using it, you are expected to register. Individual programs differ on details -- some request registration while others require it, some specify a maximum trial period. With registration, you get anything from the simple right to continue using the software to an updated program with printed manual.

Copyright laws apply to both Shareware and commercial software, and the copyright holder retains all rights, with a few specific exceptions as stated below. Shareware authors are accomplished programmers, just like commercial authors, and the programs are of comparable quality. (In both cases, there are good programs and bad ones!) The main difference is in the method of distribution. The author specifically grants the right to copy and distribute the software, either to all and sundry or to a specific group. For example, some authors require written permission before a commercial disk vendor may copy their Shareware.

Shareware is a distribution method, not a type of software. You should find software that suits your needs and pocketbook, whether it's commercial or Shareware. The Shareware system makes fitting your needs easier, because you can try before you buy. And because the overhead is low, prices are low also. Shareware has the ultimate money-back guarantee -- if you don't use the product, you don't pay for it.

Association of Shareware Professionals (ASP) Ombudsman Statement

This program is produced by a member of the Association of Shareware Professionals (ASP). ASP wants to make sure that the Shareware principle works for you. If you are unable to resolve a Shareware-related problem with an ASP member by contacting the member directly, ASP may be able to help. The ASP Ombudsman can help you resolve a dispute or problem with an ASP member, but does not provide technical support for members' products. Please write to the ASP Ombudsman at 545 Grover Road, Muskegon, MI 49442-9427 USA, FAX 616-788-2765 or send a CompuServe message via CompuServe Mail to ASP Ombudsman 70007,3536.

LIMITED WARRANTY AND LICENSE AGREEMENT

NO WARRANTIES. Users of AA-Array.DLL must accept this disclaimer of warranty: "AA-Array.DLL is supplied as is. The author disclaims all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The author assumes no liability for damages, direct or consequential, which may result from the use of AA-Array.DLL. The entire risk arising out of use or performance of AA-Array.DLL remains with the User."

CUSTOMER REMEDIES. AA-Software International's entire liability and your exclusive remedy shall not exceed the registration fee for AA-Array.DLL.

REGISTRATION AND USE. AA-Array.DLL is a "Shareware program" and is provided at no charge to the user for evaluation. Feel free to share it with your friends, but do not give it away altered or as part of another system. The essence of "user-supported" software is to provide personal computer users with quality software without high prices, and yet to provide incentive for programmers to continue to develop new products. If you find this program useful and find that you are using AA-Array.DLL and continue to use AA-Array.DLL after a reasonable trial period, you must make a registration payment to AA-Software International.

The registration fee will license one copy of the Development version of AA-Array.DLL for use on any one computer at any one time. You must treat this software just like a book. An example is that this software may be used by any number of people and may be freely moved from one computer location to another, so long as there is no possibility of it being used at one location while it's being used at another. Just as a book cannot be read by two different persons at the same time.

The registration fee will license you to distribute the Production version of AA-Array.DLL with your Visual Basic program for use on any number of computers. The registration fee will license you to use the provided source code and to distribute a modified Production version of AA-Array.DLL with your Visual Basic program for use on any number of computers. The registration fee will NOT license you to distribute a modified version of AA-Array.DLL of any kind for development purpose.

Commercial users of AA-Array.DLL must register and pay for their copies of AA-Array.DLL within 30 days of first use or their license is withdrawn. Site-License arrangements may be made by contacting AA-Software International.

DISTRIBUTION. Anyone distributing AA-Array.DLL for any kind of remuneration must first contact AA-Software International at the address below for authorization. This authorization will be automatically granted to distributors recognized by the (ASP) as adhering to its guidelines for Shareware distributors, and such distributors may begin offering AA-Array.DLL immediately (However AA-Software International must still be advised so that the distributor can be kept up-to-date with the latest version of AA-Array.DLL.).

You are encouraged to pass a copy of AA-Array.DLL along to your friends for evaluation. Please encourage them to register their copy if they find that they can use it. All registered users will receive a copy of the latest version of the AA-Array.DLL system.

AA-Array.DLL Version 1.0

Unregistered Shareware Version

Please register by contacting:

AA-Software International
12 ter Domaine Du Bois Joli
06330 Roquefort-Les-Pins, France

Tel: (+33) 93.77.50.47 **Internet: cswilly@acm.org**
Fax: (+33) 93.77.19.78 **CompuServe: 100343,2570**
X400: (C=US; A=CompuServe; P=csmail; D=ID:100343,2570)

Glossary

Simple Visual Basic Types
Persistent File (persistence)
Registered Version
User Defined Types
Shareware DLL
Development DLL
Production DLL
API

Simple Visual Basic Types

These are:

- Integer
- Long
- Single
- Double
- Currency
- String

Persistent File (persistence)

This file is created when an array should be saved between program executes. The data in the file "persists" between program runs.

Registered Version

As one of the benefits of registering, the registered version of the DLL has more features especially needed for the processional Visual Basic programmer.

User Defined Types

User Defined Types are created with the `Type` statement. For example the User Defined Type, `Foo`, is declared with the following statement:

```
Type Foo
      int as integer
      str as string
      lng as long
End Type
```

Shareware DLL

Fully working version of the Development DLL without routines for User defined types: AryOpen, ArySetElement, and AryGetElement.

Development DLL

Used by the Visual Basic programmer for debugging his/her software. Full of error trapping messages to help with your debugging. No nag screens.

Production DLL

This DLL should be distributed with your application. It is smaller and faster than the Development DLL. The error trapping messages used by the Visual Basic programmer for debugging are removed. No nag screens.

API
Application Programming Interface

AA-Array.DLL Version 1.0AA-Array.DLLAA-Array.DLLAA-ARRAY4112

Shareware

Legal Matters / Definition of Shareware
Legal Matters / Association of Shareware Professionals (ASP) Ombudsman Statement
Glossary / Shareware DLL

API

AA-Array.DLL Version 1.0 Documentation / AA-Array.DLL API Description
Glossary / API

AryOpen*

Create or Open an Extended Array / AryOpen* Declarations
Create or Open an Extended Array / AryOpen* Constant Definitions
Create or Open an Extended Array / AryOpen* Mode Bit Definitions

AryGetBounds

AA-Array.DLL API Description / Managing Array Bounds (ArySetBounds, AryGetBounds, AryCheckIndex)
Managing Array Bounds (ArySetBounds, AryGetBounds, AryCheckIndex) / ArySetBounds, AryGetBounds, and AryCheckIndex Declarations

ArySetBounds

AA-Array.DLL API Description / Managing Array Bounds (ArySetBounds, AryGetBounds, AryCheckIndex)
Managing Array Bounds (ArySetBounds, AryGetBounds, AryCheckIndex) / ArySetBounds, AryGetBounds, and AryCheckIndex Declarations

AryAbort

AA-Array.DLL API Description / Ending Use of an Array (AryClose and AryAbort)
Ending Use of an Array (AryClose and AryAbort) / AryClose and AryAbort Declarations

AryClose

AA-Array.DLL API Description / Ending Use of an Array (AryClose and AryAbort)
Ending Use of an Array (AryClose and AryAbort) / AryClose and AryAbort Declarations

AryGetStatus

AA-Array.DLL API Description / Getting Information (AryIsEmptyElement, AryGetStatus, and AryVersion)
Getting Information (AryIsEmptyElement, AryGetStatus, and AryVersion) / AryIsEmptyElement, AryGetStatus and AryVersion Declarations

registered version

AA-Array.DLL API Description / Using User Defined Types (registered version only)
Glossary / Registered Version

User defined types

AA-Array.DLL API Description / Using User Defined Types (registered version only)
Glossary / User Defined Types

register

AA-Array.DLL Version 1.0 Documentation / Benefits of Registering
AA-Array.DLL Version 1.0 Documentation / How to Register
AA-Array.DLL API Description / Using User Defined Types (registered version only)
Glossary / Registered Version