

CBTBase64.Base64 Object

This object is used to encode/decode Base64 files and strings. It does this with help from the MSXML object, which contains a node with datatype bin.base64. The object can encode/decode small strings (actually there's "no limit") and entire files of multiple gigabytes. This is done by encoding/decoding the files in several smaller blocks. The default value for the block size is 1Mb (rounded as needed for encoding/decoding).

I actually believe this is one of the fastest ways to encode/decode base64 data in Visual Basic.

Let's get it going on with the documentation:

The Base64 object exposes several methods, properties and events that make it easy for you to work with it in any (Com enabled) environment like Visual Basic, ASP and so on. They are as follows:

Properties:

LastErrorCode (*Long*)

Read-only property that returns any error code that last occurred. This property is handy when you are in an environment which doesn't support events (like the event ErrorOccured). This property can be cleared with the .ClearError method.


LastErrorDescription (*String*)

Read-only property that returns any error description that last occurred. This property is handy when you are in an environment which doesn't support events (like the event ErrorOccured). This property can be cleared with the .ClearError method.


BlockSize (*Long*)

Write-only property that sets the preferred block size when reading or writing files with the .EncodeFile or .DecodeFile methods. The file will be read or written with "chunks" of Blocksize bytes. The block size is actually rounded a bit for reading and writing to optimize for the Base64 Encoding, but will always be as close as the required size as possible. The Default size is approx. 1Mb, but tinkering with this value can result in faster or slower file operations. A block size which is set too low will result in reading or writing too many blocks, and thus slowing down because of too many file operations (required block reads/writes) and a block size which is set too high will result in higher memory usage and slower encoding or decoding of the base64 data. Good values are around .5Mb to approx 4Mb. This may vary on other systems.

Methods:

 **Decode**(ByVal strIn As String) As String


The decode method decodes data passed in strIn and returns a string that contains the decoded base64 data. This data may contain binary data!

 **DecodeFile**(ByRef strInFile As String, ByRef strOutFile As String) As Boolean

The decodefile method decodes a file, passed in strInFile to a file which is specified in strOutFile. The data is read and decoded in .blocksize bytes (approx.). When strInFile does not exist, an error event will be raised (ErrorOccured) and the error will also be available in the .LastErrorCode and .LastErrorDescription properties. The file specified in strOutFile will be overwritten if it exists, so you might want to check yourself if the file exists (either by checking it before calling this method, or in the BeforeFileOpenOut event). This method returns true if decoding of the file succeeded else it will return false.

 **Encode**(ByVal strIn As String) As String

The encode method encodes data passed in strIn and returns a string that contains the encoded base64 data. The data passed in strIn may contain binary data!

 **EncodeFile**(ByRef strInFile As String, ByRef strOutFile As String) As Boolean

The encodefile method encodes a file, passed in strInFile to a file which is specified in strOutFile. The data is read and encoded in .blocksize bytes (approx.). When strInFile does not exist, an error event will be raised (ErrorOccured) and the error will also be available in the .LastErrorCode and .LastErrorDescription properties. The file specified in strOutFile will be overwritten if it exists, so you might want to check yourself if the file exists (either by checking it before calling this method, or in the BeforeFileOpenOut event). This method returns true if encoding of the file succeeded else it will return false.

 **ClearError**

Clears the .LastErrorCode and .LastErrorDescription properties.

Events:

AfterFileCloseIn(ByVal strFileName As String)

This event is raised whenever a file that has been used for input is closed. strFileName contains the filename of the file that has been closed.

AfterFileCloseOut(ByVal strFileName As String)

This event is raised whenever a file that has been used for output is closed. strFileName contains the filename of the file that has been closed.

BeforeFileOpenIn(ByVal strFileName As String, ByRef bCancel As Boolean)

This event is raised whenever a file is about to be opened for input. strFileName contains the filename to be opened. You can cancel this by setting bCancel = true, in which case the entire encodefile of decodefile method will be cancelled.

BeforeFileOpenOut(ByVal strFileName As String, ByRef bCancel As Boolean)

This event is raised whenever a file is about to be opened for output. strFileName contains the filename to be opened. You can cancel this by setting bCancel = true, in which case the entire encodefile of decodefile method will be cancelled.

BlockRead(ByVal lngCurrentBlock As Long, ByVal lngTotalBlocks As Long, ByVal IBlockMode As enBlockMode, ByRef bCancel As Boolean)

This event is raised whenever a block has been read (and en/decoded). lngCurrentBlock contains the current block which has been en/decoded and lngTotalBlocks contains the total number of blocks that will be en/decoded. This property can be used for displaying progress (with help from a progress bar, or as a percentage for example). IBlockMode contains what has been done to the block (e.g. encoded or decoded). IBlockMode returns an enum, enBlockMode. When bCancel is set to true, the entire encodefile or decodefile method will be aborted.

ErrorOccured(ByVal lngCode As Long, ByVal strDescription As String)

This event is raised whenever an error occurred in the base64 object. lngCode contains the error number, strDescription contains the error description. Both these values are also available via the .LastErrorCode and .LastErrorDescription properties.

FileDecodeComplete(strOriginalFile As String, strDecodedFile As String)



This event is raised whenever a file has completely been decoded. strOriginalFile contains the filename of the file that has been encoded, strDecodedFile contains the filename of the decoded file.

FileEncodeComplete(strOriginalFile As String, strEncodedFile As String)

This event is raised whenever a file has completely been encoded. strOriginalFile contains the filename of the file that has been encoded, strEncodedFile contains the filename of the encoded file.

Enums:

enBlockMode

- | | |
|--|--------------------------------------|
|  b64Encode (value: 0) | : the current block is being encoded |
|  b64Decode (value: 1) | : the current block is being decoded |

Disclaimer:

This source code will be released under GPL. Please leave me a message if part of this code fit your effort in any way. Usage of this code is at own risk, the author cannot be held responsible for ... just about anything ;-)

Author: Rob Janssen (r.janssen@combat.cc)

Date: October 18th, 2004