

BinaryXtra Documentation

www.xtramania.com

BinaryXtra documentation.....	3
BinaryXtra Usage	3
About BinaryXtra.....	3
What is BinaryXtra?	3
Common guidelines.....	4
BinaryXtra object model.....	4
Creating binary data wrapper and filling it with a data	4
Using binary data wrapper.....	5
Debugging recommendations.....	6
Shockwave usage	7
How to... - FAQs about BinaryXtra	8
BinaryXtra Reference.....	9
Version history	9
Xtra-level functions, provided by BinaryXtra:	10
xtra "BinaryXtra".Create()	10
xtra "BinaryXtra".Init(bDebugMode)	10
xtra "BinaryXtra".Version().....	11
Methods and properties, provided by BinaryXtra wrapper:	12
objBinary.byte[index]	13
objBinary.DebugMode.....	13
objBinary.Failed.....	14
objBinary.LastError.....	14
objBinary.Media	14
objBinary.Picture.....	15
objBinary.Size.....	16
objBinary.Succeeded.....	16
objBinary.Allocate(newSize)	16
objBinary.Clear().....	17
objBinary.HexString()	17
interface(objBinary)	17
objBinary.ReadFromFile(fileName)	18
String(objBinary).....	18
objBinary.WriteToFile(fileName)	18
Contacts.....	20

BinaryXtra documentation

BinaryXtra Usage

About BinaryXtra

Several words about what BinaryXtra is and several thoughts about where to use BinaryXtra.

What is BinaryXtra?

BinaryXtra is an extension of Macromedia Director of type 'scripting xtra'. It extends the capabilities of Lingo (Director's scripting language) with ability to handle general binary data.

BinaryXtra provides a special type of Lingo variable designed to hold and handle binary data.

- Allocating in memory a new binary data block;
- Direct access to the binary data via 'byte[]' indexed property;
- Initializing itself with a file contents;
- Writing binary data into a file;
- Initializing itself with a media of a cast member;
- Setting binary data as a cast member's media;
- Initializing itself with a picture data of a cast member;
- Setting binary data as a cast member's picture;

BinaryXtra is supported by VbScriptXtra and ADOxtra. Typecasting routines of these xtras use BinaryXtra when necessary. This allows both ADOxtra and VbScriptXtra read and write BLOB data into database.

BinaryXtra is safe for Shockwave, so it can be used together with ADOxtra to dynamically download or upload media of Shockwave movie cast members.

Common guidelines

Basic things you have to know to start using BinaryXtra.

BinaryXtra object model

BinaryXtra is organized similar to ADOxtra and VbScriptXtra. BinaryXtra implements several xtra-level methods and binary data wrapper. Xtra level methods provide basic xtra initialization, version information and binary wrapper creation.

Binary wrapper provides the main functionality of the xtra. At first, it holds the binary data. Also it provides methods and properties for accessing binary data.

Creating binary data wrapper and filling it with a data

There are several ways how binary wrapper instance can be created. It may be created either directly by Create xtra-level method or indirectly by other xtras (like ADOxtra or VbScriptXtra).

Following line creates a new empty binary wrapper:

```
binary=xtra("BinaryXtra").Create()
```

Use Size property to check the size of the wrapped data in bytes:

```
put binary.Size
```

It will return zero for new empty binary wrapper.

Sample below demonstrates indirect creation of the wrapper by VbScriptXtra:

```
rst.Open("SELECT BLOB_field FROM SomeTable")
```

```
binary=rst.fields["BLOB_fields"].value
```

Note: for this sample to work BinaryXtra has to be loaded (exists in Xtras folder). This sample creates a new binary wrapper and fills it with the data contained in the database.

There are several other ways to put actual data into a binary wrapper. Use ReadFromFile method to put the contents of file into a wrapper:

```
binary.ReadFromFile("SomeFilePathName")
```

```
put binary.Size
```

Use Media property to put the media of any cast member into the wrapper:

```
binary.Media=member("SomeMember").media
```

```
put binary.Size
```

Use Picture property to put the picture data into the wrapper:

```
binary.Picture=member("SomeMember").picture
```

```
put binary.Size
```

Also binary wrapper can allocate a piece of memory, which could be filled by Lingo script directly byte by byte (array of bytes):

```
binary.Allocate(1000)
repeat with i=1 to binary.Size
    binary.byte[i]=i
end repeat
put binary.Size
```

Using binary data wrapper

Once binary wrapper is created, it may be used to directly access the bytes of data. It may be used to apply binary data as a media or picture data. Also the binary contents of the wrapper can be written into a file or passed to other xtra.

Sample below demonstrates using binary wrapper to write the media of member into a BLOB field of a database:

```
binary=xtra("BinaryXtra").Create()
binary.Media=member("SomeMember").media
rst.Open("SELECT BLOB_field FROM SomeTable")
rst.fields["BLOB_fields"].value=binary
rst.Update()
```

Use WriteToFile method to create a file and write the binary data into it:

```
binary.WriteToFile("SomeFilePathName")
```

Use Byte[] indexed property to directly access bytes of the binary data:

```
repeat with i=1 to binary.Size
    put binary.byte[i]
end repeat
```

Binary wrapper allows to convert binary contents into a string using String method: put string(binary) Note: this conversion treats the binary data as zero terminated string, so the whole binary contents will be output as string only when there are no zeros in binary data.

There is also a HexString method which allows to output binary data in a hexadecimal text form.

Debugging recommendations

Every BinaryXtra wrapper instance has internal last error flag and error description. The last error flag is cleared before any access to the binary data. If property access or method call failed or another error happened, this flag is raised. So you may detect whether last call completed successfully. Use `object.Failed` or `object.Succeeded` properties to check whether the last call was successful. If an error happened you may see its description using `object.lastError` property. Also you may adjust BinaryXtra wrapper objects to output its `lastError` directly to the Message window every time error happens. Set `object.DebugMode` to `true` (1) to do this. By default, BinaryXtra wrappers created by `Create` call inherit the xtra's default value for `debugMode`. You may change this default with `Init` xtra-level method.

Note: While you are just investigating BinaryXtra capabilities it is better to set `DebugMode` by default, to ensure you always know if something goes wrong.

Shockwave usage

To use BinaryXtra from Shockwave several extra steps are required:

At first you have to place the BinaryXtra package (certified by Verisign) on your web site. Otherwise you may use package available at:

<http://www.xtramania.com/package/BinaryXtra.w32>.

Then you have to modify xtrainfo.txt file located in you Director installation folder. This file provides the master list of xtras known to Director. Add following lines to the end of this file: [#nameW32:"BinaryXtra.x32", #info:"<http://www.xtramania.com/>", #package:"<http://www.xtramania.com/package/BinaryXtra>"]

Replace the #package values to the location on your web server where package file resides. Note: file and folder names in URL may be case sensitive.

After modifying this file you may run Director, open the movie you are going to use from Shockwave and then invoke the Modify/Movie/Xtras dialog. Find (or add) the BinaryXtra entry and place the check mark near "Download if needed" and "Include in projector". Director will check whether package file is available at this moment. Then you may save the movie as Shockwave and try it.

How to... - FAQs about BinaryXtra

There were no FAQs yet, sorry...

BinaryXtra Reference

Version history

Information about how BinaryXtra is growing.

October 21st, 2001

Version 1.00.000r03 The first public release. Xtra is available to download.

Xtra-level functions, provided by BinaryXtra:

Init(bDebugMode) - Sets debugging mode for newly created binary wrapper instances.

Create() - Creates new empty binary data wrapper instance.

Version() - Returns the version of BinaryXtra.

xtra "BinaryXtra".Create()

Syntax `objBinary=Create(xtra"BinaryXtra")` or
 `objBinary=xtra("BinaryXtra").Create()`

Returns Object: returns newly created empty binary data wrapper instance.

Description This method is the basic starting point of using BinaryXtra. It allows to directly create a new BinaryXtra wrapper for binary data. This wrapper can be used on its own and also passed to VbScriptXtra or ADOXtra as a container for binary data.

Sample `objBinary=xtra("BinaryXtra").Create()`
 `objBinary.media=member("SomeMember").media`
 `objBinary.WriteToFile(the moviePath & "SomeFile.bin")`

xtra "BinaryXtra".Init(bDebugMode)

Syntax `bSuccess=Init(xtra"BinaryXtra",bDebugMode)` or
 `bSuccess=xtra("BinaryXtra").Init(bDebugMode)`

Parameters `bDebugMode` - a Boolean value specifying whether to set debug mode for newly created wrapper instances. Note: the value of this param affects wrapper instances created by further calls to `Create` method. When debug mode is set, wrapper instance outputs all information about errors in Messages window.

Returns `true`.

Description Sets debugging mode for newly created binary wrapper instances. Debugging mode is highly recommended while investigation what BinaryXtra can do, since it will output all error description information right in Messages window. There is no need to set debug mode in release versions. You may safely call Init several times to enable or disable debug mode default setting.

xtra "BinaryXtra".Version()

Syntax `strVersion=Version(xtra"BinaryXtra")` or
 `strVersion=xtra("BinaryXtra").Version()`

Returns String with version info in a form: "BinaryXtra.1.00.000"

Description Returns the version of BinaryXtra

Methods and properties, provided by BinaryXtra wrapper:

Binary data wrapper is a key component of BinaryXtra. It is used to hold any binary data.

General binary data support methods and properties:

- `objBinary.Allocate(newSize)` - allocates a block of memory of the specified size.
- `objBinary.byte[index]` - indexed property for direct access to the bytes of binary data.
- `objBinary.Clear()` - clears any current wrapper contents.
- `objBinary.Size` - the size in bytes of the contained binary data.
- `String(objBinary)` - returns the text representation of the binary data.
- `objBinary.HexString()` - returns the hexadecimal text representation of the binary data.

File input/output support methods:

- `objBinary.ReadFromFile(fileName)` - initializes the wrapper with a binary contents of the specified file.
- `objBinary.WriteToFile(fileName)` - saves the contained binary data into the specified file.

Media and picture support properties:

- `objBinary.Media` - allows storing cast members' media in a binary wrapper
- `objBinary.Picture` - allows storing cast members' picture data in a binary wrapper

Error handling properties:

- `objBinary.DebugMode` - gets or sets debugging mode of the wrapper instance.
- `objBinary.Succeeded` - returns true if previous wrapper access to the binary data has succeeded.
- `objBinary.Failed` - returns true if previous wrapper access to the binary data has failed.
- `objBinary.LastError` - returns the last error description if any.

Autodocumentation feature:

- `interface(objBinary)` - returns a string with quick help information.

Overridden standard Lingo commands:

- `put objAuto` - puts a brief information about wrapper and wrapped data.

objBinary.byte[index]

- Syntax** `objBinary.byte[index]=newByteVal`
 `byteVal=objBinary.byte[index]`
- Parameters** `Index` - an integer 1-based offset at the binary data. The first byte has offset 1, the last byte has offset `Size`.
- Gets** Integer value in the range 0-255 - the byte at offset `index` of the binary data contained in the wrapper.
- Sets** Integer value in the range 0-255 - the byte to be put at offset `index` of the binary data contained in the wrapper.
- Description** Indexed property allows direct access to the binary data contained in the wrapper. To use this property binary wrapper has to contain something. New or empty binary wrappers do not contain any data, so using this property will generate a 'Index out of bounds' error. See debugging for more info on error reporting mechanism.

objBinary.DebugMode

- Syntax** `bDebugMode=objBinary.DebugMode`
 `objBinary.DebugMode=bDebugMode`
- Gets** Boolean value, which indicates whether wrapper currently in debug mode.
- Sets** Boolean value. Use `true` to set debug mode. Use `false` to clear debug mode of the wrapper instance.
- Description** BinaryXtra wrapper object supports special debugging mode. While the debug mode is set wrapper instance outputs any error messages directly to the Message window every time error happens. By default, BinaryXtra wrappers created by `Create` call inherit the xtra's default value for `debugMode`. You may change this default with `Init` xtra-level method.

Wrappers created by other xtras (ADOxtra or VbScriptXtra) inherit DebugMode setting from objects that created the instance.

Note: While you are just investigating BinaryXtra capabilities it is better to set Debug Mode by default, to ensure you always know if something goes wrong.

objBinary.Failed

Syntax `bResult=objBinary.Failed`

Returns Boolean value indicating whether last attempt to access wrapped binary data has failed.

Description Use `Failed` property to check whether error occurred.
See Debugging for related information.

objBinary.LastError

Syntax `strErrorDescription=objBinary.LastError`

Returns String with last error's description or empty string if the last call was successful.

Description Use `LastError` property to get the description of error occurred. Use debug mode to automatically get error descriptions in Messages window.
See Debugging for related information.

objBinary.Media

Syntax `objBinary.Media=member("SomeMember").Media`
`member("SomeMember").Media=objBinary.Media`

-
- Gets** Media Lingo value which can be used to assign a new media to a cast member. This value is populated by the binary data contained in the wrapper.
- Sets** Media Lingo value which is stored as binary data inside binary wrapper.
- Description** Use `Media` property to represent a binary data contained in the wrapper as media data which can be assigned to any cast member as its media. Assigning data to this property allows binary wrapper to fill itself with a binary media data.
- Note: media data format is internal for Macromedia Director, it is not documented and it is not guaranteed to be the same across different versions of Macromedia Director. BinaryXtra knows nothing about internal structure of the media data.

objBinary.Picture

- Syntax** `objBinary.Picture=member("SomeMember").Picture`
`member("SomeMember").Picture=objBinary.Picture`
- Gets** `Picture` Lingo value which can be used to assign a new picture to a bitmap cast member. This value is populated by the binary data contained in the wrapper.
- Sets** `Picture` Lingo value which is stored as binary data inside binary wrapper.
- Description** Use `Picture` property to represent a binary data contained in the wrapper as picture data which can be assigned to any cast member as its media. Assigning data to this property allows binary wrapper to fill itself with a binary picture data.
- Note: picture data format is internal for Macromedia Director, it is not documented and it is not guaranteed to be the same across different versions of Macromedia Director. BinaryXtra knows nothing about internal structure of the picture data.

objBinary.Size

Syntax `dataSize=objBinary.Size`

Returns Integer value which indicates the size in bytes of the wrapped binary data.

Description Use the `Size` property to determine the size of the binary data.

objBinary.Succeeded

Syntax `bResult=objBinary.Succeeded`

Returns Boolean value indicating whether last attempt to access wrapped binary data has completed successfully.

Description Use `Succeeded` property to check whether no error occurred.
See `Debugging` for related information.

objBinary.Allocate(newSize)

Syntax `objBinary.Allocate(newSize)`

Parameters `newSize` - Integer value which indicates a requested size for binary data.

Returns VOID

Description Allocates a new binary data block of the requested size and fills the allocated memory with zeros. This method destroys any previous contents of the wrapper instance. Use this method if you wish to generate binary data via Lingo in array-like manner. See `byte[]` indexed property for related information.

Note: there is no need to call this method before initializing wrapper in any other way.

Sample `binary.Allocate(1000) repeat with i=1 to binary.Size
binary.byte[i]=i end repeat put binary.Size`

objBinary.Clear()

Syntax `objBinary.Clear()`

Returns VOID

Description Use this method to clear any current binary contents and return the wrapper into initial state.

objBinary.HexString()

Syntax `put objBinary.HexString()`

Returns String value with hexadecimal representation of the binary data contained in the wrapper.

Description Use this method to get hexadecimal representation of the binary data.

interface(objBinary)

Syntax `put interface(objBinary)`

Returns String with a self descriptive information about properties and methods implemented by BinaryXtra wrapper.

Description Use this method to get the quick info about BinaryXtra wrapper interface.

objBinary.ReadFile(fileName)

Syntax objBinary.ReadFile(fileName)

Parameters fileName - string value which indicates which file to read.

Returns VOID

Description Use ReadFromFile method to initialize binary wrapper with the contents of the specified file.

Note: disabled when run as Shockwave movie.

Sample For example, you may store the media of some cast member in external

file and later retrieve it and assign to the member:

```
objBinary=xtra("BinaryXtra").Create()
```

```
objBinary.media=member("SomeMember").media
```

```
objBinary.WriteToFile("SomeFile") objBinary.Clear()
```

```
objBinary.ReadFile("SomeFile")
```

```
member("SomeMember").media=objBinary.media
```

String(objBinary)

Syntax put String(objBinary)

Returns String representation of the binary data.

Description Use this method if binary wrapper contains text data. Note, if binary data contains zeros, then the returned string will be truncated by the first zero byte.

objBinary.WriteToFile(fileName)

Syntax objBinary.WriteToFile(fileName)

Returns VOID

Description Use `writeToFile` method to save contained binary data into the specified file.

Note: disabled when run as Shockwave movie.

Sample For example, you may store the media of some cast member in external file and later retrieve it and assign to the member:

```
objBinary=xtra("BinaryXtra").Create()  
objBinary.media=member("SomeMember").media  
objBinary.WriteToFile("SomeFile") objBinary.Clear()  
  
objBinary.ReadFromFile("SomeFile")  
member("SomeMember").media=objBinary.media
```

Contacts

If you wish to contact xtras' author, you may contact me by e-mail: author@xtramania.com

Best regards, Eugene Shoustrov