Xpk

COLLABORATORS			
	TITLE : Xpk		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 9, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Xpk

1.1 Xpk V1.00

```
                  Xpk V1.00 General Information:

* Blitz Basic II library number         : #178
* Library size when linked to executable: 916 bytes
* Number of commands                    : 14
* Ressources automatically freed at end : No

NInitXpk() must be put before any other Xpk functions
or you will enjoy BIG crashs.

  Commands summary:


               NGetNextPacker
               Function (String)

               NGetPackerNumber
               Function (Long)

               NInitXpk
               Command (Boolean)

               NPackerDescription
               Function (String)

               NPackerInfo
               Statement

               NPackerLongName
               Function (String)

               NPackerName
               Function (String)

               NPackFile
               Function (Long)

               NPackMemToFile
```

```
                Function (Long)

                NScanPackersList
                Statement

                NUnPackFile
                Fonction (Long)

                NUnPackFileToMem
                Fonction (Long)

                NXpkBufferLength
                Function (Long)

                NXpkFileLength
                Function (Long)
```

1.2 ninitxpk

```
  SYNTAX
result.l = NInitXpk

  COMMAND
Init all the Xpk environnement for  later  use.  You  must  put  this
functions  on  top  of  your  source code if you want to use the NXpk
commands. This command try to open the xpkmaster.library V3.0+. If
result is NULL, the library can't be opened so be sure to test the
result and disable all your XPK calls if it fails (or quit the program).
```

1.3 nunpackfile

```
  SYNTAX
res.l = NUnPackFile(&SourceFile$, &DestFile$, &Password)

  COMMAND
Simply unpack an XPK Packed file to an other file. Is the file
is password protected, you must specify it.

If res = 0, the unpack has failed.
```

1.4 npackfile

```
  SYNTAX
res.l = NPackFile(&SourceFile$, &DestFile$, &PackerName, &Password)

  COMMAND
Pack the given file to an other file using the specified packer and
password.
```

If res = 0, the pack has failed.

1.5 nunpackfiletomem

```
  SYNTAX
*MemLocation = NUnPackFileToMem(&SourceFile$, MemType, &Password)

  COMMAND
```

Unpack the specified file to a memory zone which will be allocated automatically. The fonction return the memory addresse where the file is depacked. You must use the XpkBufferLen() fonction to know how big is the buffer. you must free the buffer manually after finish to use it with the following command:

```
          FreeMem_ *MemLocation, XpkBufferLen
```

MemType allow you to specify which memory should be used for the buffer (CHIP or FAST mem)

If *MemLocation = 0, the unpack has failed.

1.6 npackmemtofile

```
  SYNTAX
res.l = PackMemToFile(*MemAddr, MemLen, &DestFilename$, &Packer$, &Pass$)

  COMMAND
```

Pack the memory starting from *MemAddr, of MemLen to DestFileName$ with the Packer$ and Pass$.

If res = 0, the pack has failed.

1.7 nxpkbufferlength

```
  SYNTAX
BufferLen.l = NXpkBufferLength

  FUNCTION
```

Return the buffer length of the last NUnFileToMem decompressed file. This is used to allow you to free the buffer after use as the xpk package and Blitz doesn't free it.

Warning ! Buffer length is different (bigger) than unpacked file length because it create an header for handling the file. So be sure to use this buffer length for free the memory and not the NXpkFileLen() command.

1.8 nxpkfilelength

```
  SYNTAX
BufferLen.l = NXpkFileLength

  FUNCTION
Return the file length of the last NUnPackFileToMem() function, to
know the exact unpacked size of the file.

Warning ! Unpacked file length is different than buffer lenght
allocated by Xpk when Unpacking to memory !
```

1.9 npackerinfo

```
  SYNTAX
NPackerInfo(Name$, Efficiency)

  STATEMENT
Search the informations from a specified packer. Use the following commands
to get special informations: NPackerName(), NPackerLongName(),
NPackerDescription().
```

1.10 npackername

```
  SYNTAX
Text$ = NPackerName

  FUNCTION
Return the packer name got with the NPackerInfo() function.
```

1.11 npackerlongname

```
  SYNTAX
Text$ = NPackerLongName

  FUNCTION
Return the packer long name got with the NPackerInfo() function.
```

1.12 npackerdescription

```
  SYNTAX
Text$ = NPackerDescription

  FUNCTION
Return the packer description got with the NPackerInfo() function.
```

1.13 nscanpackerslist

```
  SYNTAX
NScanPackersList

  STATEMENT
Scan the full packers list. Use NGetPackerNumber() and NGetNextPacker()
to access the datas collected with this scan.
```

1.14 ngetpackernumber

```
  SYNTAX
Number.l = NGetPackerNumber

  FUNCTION
Return the number of packers available on the system. This value must
be got before by calling the NScanPackersList() fonction before.
```

1.15 ngetnextpacker

```
  SYNTAX
Text$ = NGetNextPacker

  FUNCTION
Return the next packer name found with a NScanPackersList() call.
```