# Introduction


### Purpose

LZAPI is an advanced Interface for the Programmer who wants to work with Archived Files in his / her Application.

Archives are Files that contain other Files, typically the Files inside the Archives are compressed. Archives help their Users to group Files together and reduce their Size, so Handling and Copying of Files will be easier.

A number of Programs are available to work with compressed Files, and as different as the Programs are the File Formats. The Extension usually shows which Program generated the File, and which Program will be needed to handle it, so the filename usually ends with ".LZH", ".ARJ", ".ARC", ".ZIP", "ZOO" or ".RAR" depending on the Program that created it.

Like the Name "Archiv" implies, an Archiv is NOT your Applications main workfile. It is an compressed and archived Version of your Workfiles, that means, you can only work with them after extracting them.

### Packaging Parts

The LZAPI-Package Consists of three Parts:

1. The LZAPI.DLL Archive File Handing routines, that do mainly the Same as PKZIP or ARJ inside of a DLL.

2. The LZRES.OBJ-File that helps you to hold Bitmaps and other Resource Data inside your EXE-File and to decompress it into Memory at Runtime.

3. The LVIEW Demo for LZAPI that shows you the powerfull Archive-File-Handling features.

### License

A License for LZAPI will cost you 99 US $ / 150 DM and includes the right to give all Archiver-DLLs included with this package with your program without any payment per package. See Licensing Terms for details.

### The Programmers View

For the operating System, the archived file is a File like any other. You may list it with "DIR", but you can't look "into" it.

To work with the Archive in your Program, you'd normally have to use the Program intended to be used for this Purpose:

| File Extension | Program for Creation | Program for Extraction | Author/Copyright |
|---|---|---|---|
| ARC | PKARC | PKXARC | PKWARE, Inc. |
| ARJ | ARJ | ARJ | Robert K Jung |
| LZH | LHA | LHA | Haruyasu Yoshizaki |
| RAR | RAR | RAR | Eugene Roshal |
| ZIP | PKZIP | PKUNZIP | PKWARE, Inc. |
| TAR | TAR | TAR | Unix Tool (GNU) |
| ZOO | ZOO | ZOO | Rahul Dhesi |

All these Programs are DOS-Commandline-Oriented Programs using a Command-String to control their behaviour. So your Program will have to deal with DOS-Boxes, command-Interpreters and the differently defined Command Arguments for the Packers.

In addition your User may not have one of the Packers, so you may have to send it with your Program, so you'll have to pay the Licensing fees.

For some of the File-Formats, other solutions are available, The Freeware-ZIP-Packer-Soirces from Info-Zip are one example, but most of them are also DOS-Commandline Based Applications.

### Programmers view with LZAPI

The LZAPI Dynamic Link Library brings to you a common Archiver-API that gives you an abstract View to the Archiv, independend of the physical File-Format.

The most important Decompression and Listing-terms may be done with DLL-Solutions included in LZAPI. No DOS-Boxes will be needed.

| Extension | DLL | Infos |
|---|---|---|
| ARC | WARC | Listing, Extraction, Creation, Appending, 16-Bit-API Only<br>32-Bit Version scheduled for April 1996 |
| ARJ | WUNARJ | Listing and Extraction |
| LZH | WLHA | Listing, Extraction, Creation, Appending |
| ZIP | WUNZIP | Listing and Extraction |

So you may Display the Contents of the above Archive Types and expand all or selected Files without any hazzle.

### Abstract DOS-Box usage

For other purposes, like adding Files to ARJ-Archives, LZAPI will automatically open a DOS-Box to start the DOS-Archiver needed. This will be done using the same API-Calls as for the internal DLLs. In most cases your Program will not even notice it.

Filter-Routines built into the LZAPI.DLL will convert the Directory-Listings of the Achivers to the LZAPI.-Standard-Format. All types of command-Strings will automatically be matched.

### Powerfull DEMO-Source

The Source-Code of sophisticated LZVIEW-Application contained as Executable in the Shareware-Version is available as extended Source-Code-License as OWL 2.5 Borland C++ Version 4.5 Source. This

gives you one of the most exciting ZIP-Shell applications as base to start from for your own Application development. (Note: The DAVINCI Graphics Filters are a seperated Product and have to be registered individually). See the Licensing Terms for more Details and Prices.

A second, smaller DEMO-Program is included for Pascal-Programmers and shows up how to use the WLHA-Library directly to manipulate LZH-Archives.

Please Use the ">>"-Button to go to the next Help Page.

# How to Print the Documentation

LZAPI includes the PRTALL extension to WinHelp.

This means, you can start a Printout of all the Help-File Pages, page by page.

Start PRTALL Printing Technology

You may also use our *Help to Rtf*-Converter which is capable of converting the HLP-File into a printable RTF-Document file. *Help to Rtf* is available for Download as HLPRTF.ZIP on Compuserve WINSDK.

Please Use the ">>"-Button to go to the next Help Page.

# What's new in Version 2.0


Version 2.0 if LZAPI is a major upgrade of our Software, so you may think now "what the hell have they done to it?"

### Win32 Support

All the LZAPI-Files are now available as 32-Bit-Versions useable under Windows NT and Windows 95. They may also be used with Win32s, however there are some restrictions concerning the use of DOS-Based archivers with Win32s.

The 32-Bit-Versions supports the use of long filenames.

The MFILE-API supported by the 16-Bit DLL WLHA.DLL is not available in the 32-Bit-Version.

### Compressor Speedup

One speed-Problem with LZAPI in Version 1.x resulted from the limitation to one Wildcard-Filename per LAAppend-Call. Most compressors need to copy a whole Archiv to add or freshen a File, so things became very slow when Archives were big.

The solution to this Problem was a change in the definition of the lpstrWildcards-Entry of the control-Structure thus allowing you to specify as many Files as you need in *one* call to LAAppend.

In addition we added the LAF_ALLOWGROWEXIST-Flag for the use with the Flags-Entry of the control-Structure. This allows an existing archiv to grow by changing Files and marking them as deleted instead of copying (packing) the whole Archiv. This feature is currently available for ZOO and ZIP-Archives only, because only their File-Formats allow us to do it this way. This feature is especially interesting when adding some small files to a huge Archiv.

### Enhanced Callback-Support

We've added some fields to the LACTL control-Structure and some Messages to the Callback-Function filling this new Fields with concrete Informations:
LAM_NEXTFILE - Gives you the Filename of the File currently processed (Appended/Extracted)
LAM_PERCENTOFFILE - Gives you a Chance to update a Progress-Indicator (GAuge) in your Application.
LAM_MAPNAME - Allows to change the Name of the extracted file during the Extraction of a ZIP-Archiv.
These Messages are trensfered to the application-supplied Callback-Function in Addition to the LAM_PEEK Message known in LZAPI Version 1.x.

### ZOO Files

ZOO-Files can now be processed on DLL-Level via the WZOO.DLL / WZOO32.DLL.

### Optimizations

Several DLLs have been optimized in Size and Speed. Especially fast are the new WZIP/WUNZIP DLLs, where the 32-Bit-Versions are the fastest of all supported DLLs.

### LZANI removed

Support for LZANI.DLL has been removed from LZAPI for it is now a separate Product with many nice new Features like JPEG-Decompression and Multimedia Viewer Support (Licensing fee is 20 $ when using SWREG). Download JPGANI.ZIP from the WINSDK-Forum on Compuserve or contact us for further Information.

### Updates

Update-Prices are listed in the Licensing-Topic.

# Tutorial: Using LZAPI in a C/C++-Program

**Step 1:**

include LZAPI.H in your Source File

**Step 2:**

Run IMPLIB and include LZAPI.LIB in your Project(link) File.

For Microsoft Visual C++ 32-Bit use the LIB-Files included in the distribution archiv and installed into directory MSVC, as the Microsoft 32-Bit linker is unable to use import library definitions.

**Step 3:**

For File Compression purposes you use the LAAppendDirect-Function.

LAAppend creates or appends to an Archiv:

An Example:

#include <lzapi.h>

```
main()
{ LAAppendDirect(  NULL,                // Should be a Window-
Handle...
                   "C:\\TEST.LZH",      // The Archiv-Name
                   NULL,                // Source Path (not needed)
                   "C:\\AUTOEXEC.BAT",  // Next File to Add
                   LAF_SHORTNAMES,      // Flags
                   NULL);               // Callback Function
}
```

We recommend to use the LAF_SHORTNAMES - Flag.

Please Use the ">>"-Button to go to the next Help Page.

# Basic Operation

The Basic-Operations supported by LZAPI for every Archiv-Type supported are:

| | | |
|---|---|---|
| List | LAList | Lists the Archive Contents into a given String-Buffer. |
| Extract | LAExtract | Extract one or more Files from an Archive. |
| Append | LAAppend | Add one or more Files to an Archive. |
| Delete | LADelete | Delete one or more Files from an Archive |
| Test | LAIsArchiv | Test for the Type of Archiv the Archiv (for Example in an SFX/EXE) |

All of these Functions will return an error-Code of type LAERR.

To inform LZAPI about Parameters (Like Filenames, Wildcards etc.). there are two possibilities (Like you may know from the Windows-API-Functions CreateFont and CreateFontIndirect).

The more flexible way of Operation is to use the LACTL-Structure. This means you will declare a LACTL-Variable and fill out the Variables contained in it. Afterwards you will call LAList, LAExtract, LAAppend or LADelete.

The more spontainous way is to use the Direct-Parameter-Versions LAListDirect, LAExtractDirect, LAAppendDirect and LADeleteDirect.

Note that the Indirect calling Method will give you a more Version-Independend Method, because we will be able to support new Variables in the LACTL-Structure without you recompiling your Program. These new Features will not be accessible when using the Direct-Calling Versions.

# LACTL: The Control Structure

The LACTL Control-Structure defines all the Basic Functionality for calls to LAAppend, LAList, LAExtract and LADelete. A Pointer to the LACTL-Structure will also be transfered to any Application-Given Callback Function.

The following code is used in the LZAPI.H - Headerfile to declare the Control-Structure:

```
// ----------- Define Control Structure ------------------
typedef struct tagLACTL {
        DWORD        lStructSize;
        HWND         hwndOwner;
        LPCSTR       lpstrArchivFile;
        LPCSTR       lpstrWildcards;
        LPCSTR       lpstrPath;
        DWORD        Flags;
        LAERR (CALLBACK* lpfnCallback)(int, struct tagLACTL far *);

        // -------- Listing-Function only ------------------------
        LPSTR        lpstrListing;

        // -------- Application-Reserved Fields ------------------
        LPARAM       lParam;
        void FAR*lpVoid;
        int          iCounter;

        // -------- New in Version 1.3 ---------------------------
        short        ArchiverType;

        // -------- New in Version 2.0 ---------------------------
        LPCSTR       lpstrProcessedFile;
        UINT         uPercentOfFile;
        UINT         uPercentOfArchiv;

        DWORD        Reserved[10];

        } LACTL, FAR *LPLACTL;
```

| | | |
|---|---|---|
| lStructSize | DWORD | Specifies the length of the Structure in Bytes. This member is filled by the Application Program.<br><br>You have to set this Value to sizeof(LACTL) so the DLLs can see which API-Version you are using. |
| hwndOwner | HWINDOW | Identifies the Window-Handle that owns the MessageBoxes and Dialogs of LZAPI. The use of the Applications Main Window is recommended. This member is filled by the Application Program. Only Programs that do not have any Windows may use NULL for this Parameter. |
| lpstrArchivFile | LPCSTR | Points to a Buffer that contains the Name of an Archiv-File (for Example "F:\\DOWNLOAD\\LZAPI.LZH"). This Member relates to the current Directory of a DOS-Archiver Program. |

This Member is filled by the Application Program.

| | | |
|---|---|---|
| lpstrWildcards | LPCSTR | Points to a Buffer that contains the Filename to the File beeing added, deleted or extracted. Wildcards (Like "*.TXT") may be used.<br><br>If lpstrWildcards is NULL, this means to process all Files like "*.*".<br><br>Wildcards may also contain a Path-Name relative to the Directory-Name in "lpstrArchivFile", but they must not contain ".\" and "..\"-references.<br><br>During Calls to the LAAppend-Function you may also List several Wildcard-Strings Like "*.DOC *.DOT *.BMP".<br><br>This Member is filled by the Application Program. |
| lpstrPath | LPCSTR | Points to a Buffer that contains the Directory-Name to be used for Append and Extract-Operations. NULL means to use the currently set Directory.<br><br>This Member is filled by the Application Program. |
| Flags | DWORD | Flags specifying additional behaviour. |

LAF_SHORTNAMES
Use short-Filenames without a Directory name. This flag is recommended, because long Filenames are treted differently by the different archiver Programs.

LAF_ALLOWGROWEXIST
Used when adding files to an existing ZIP-or ZOO-Archiv. Without the LAF_ALLOWGROWEXIST-Flag the whole Archiv will be copied to a temporary File before adding new Files.

With LAF_ALLOWGROWEXIST, the new Files will be added to the existung Archiv, so speed is much higher. Note that the standard-Technology has better data security. Archiv-Files may also grow larger than neccessary when using LAF_ALLOWGROWEXIST

LAF_RECURSEINTO
Used when adding files to an existing ZIP-or ZOO-Archiv, so Files matiching the Wildard-Specification are also scanned in Subdirectories.

This Member is filled by the Application Program.

| | | |
|---|---|---|
| lpfnCallback | LACALLBACK | Address of a Callback-Function to support "Multitasking" in Windows 3.1 or NULL.<br><br>This Member is filled by the Application Program. |

| | | |
|---|---|---|
| lpstrListing | LPSTR | Points to an Application-Supported Buffer for the Directory contents when performing an LAList-Operation.<br><br>All Directory Listings will be presented in the same Format. |
| lParam | LONG | For the private use thru the Application |
| lpVoid | void * | For the private use thru the Application |
| iCounter | int | For the private use thru the Application |
| ArchiverType | short | Type of Archive that shall be handled. ArchiverType may be TY_NONE to force LZAPI to look for the Filename and the File Contents (If it's available) to find out itself about the Archiver Type. But when creating an Archiv with a non-Standard Extension, you'll have to Inform LZAPI in this field about the Archiver Type you want to use:<br><br>TY_NONE 0<br>TY_LZH 1<br>TY_ARC 2<br>TY_ARJ 3<br>TY_ZIP 4<br>TY_RAR 5<br>TY_TAR 6<br>TY_ZOO 7 |

The Following Fields have been added in Version 2.0 of LZAPI:

| | | |
|---|---|---|
| lpstrProcessedFile | LPSTR | Filename of File being Processed. Only valid during Processing of LAM_NEXTFILE-Calls to the Callback-Function. |
| uPercentOfFile | UINT | Percentage (0..100) of Data processed from currently extracted/appended file. Only valid during Processing of LAM_PERCENTOFFILE-Calls to the Callback-Function. |
| uPercentOfArchiv | UINT | Percentage (0..100) of Data processed from currently extracted Archiv. Only valid during Processing of LAM_PERCENTOFFILE and LAM_NEXTFILE-Calls to the Callback-Function. This Member is only maintained by some Archiver-DLLs. |
| Reserved | 40 Bytes | 40 currently unused Bytes that must be 0. |

All fields of the LACTL-Structure that are not needed by the current Process should be set to Zero. It's good Practice to use a `memset` alias `FillChar` -Call to set all the Bytes to zero.

# LAList Function

The LAList-Function lists the Files in an Archiv into a given Listing-Buffer.

LAERR FAR PASCAL LAList (LPLACTL lpCtl);

lpCtl is a Pointer to an Application-Supported LACTL-Structure control-Structure.

The lpstrListing-Member of the LACTL-Structor should point to a Listing-Buffer that can be used to receive the Listing. The Listing will be contained in ONE string with only one delimiting Zero. Every Line in the Listing will be seperated thru a '\n' - Character (ASII-Value 10). The Columns in the Listing will be seperated thru '\t' - Characters (ASCII-Value 9).

The Columns will be listed in the Following order:

Filename
uncompressed Length
compressed Length
Compression ratio
File Date
File Time

Example:

The following small example Procedure copies the Listing of Archiv C:\\LZAPI.LZH into a Listbox.

```
void FillListboxWithArchivContents(HWND HDialog)
{LACTL  ctl;
 LPCSTR pszNext;
 memset(ctl, 0, sizeof(ctl));
 ctl.lStructSize = sizeof(ctl);
 ctl.hwndOwner = HDialog;
 ctl.lpstrArchivName = "C:\\LZAPI.LZH";
 ctl.Flags = LAF_SHORTNAMES;
 ctl.lpstrListing = (LPSTR) malloc(32767);

 LAList( &ctl );

 for (pszNext = strtok(ctl.lpstrListing, "\n");
      pszNext;
      pszNext = strtok(NULL, "\n")) {
     SendDlgItemMessage( HDialog, ID_LIST, LB_ADDSTRING, 0, (LPARAM)
pszNext);
 }

 free(ctl.lpstrListing);
}
```

Note : You may want to use LB_SETTABSTOPS.

The Format for the Listing defined above is very usable to fill Multiline-Edit-Elements or use The OWL-Class TListView to fill the Elements without any further activity.

# LAExtract Function

The LAExtract-Function extracts one or more Files from an Archiv.

LAERR FAR PASCAL LAExtract(LPLACTL lpCtl);

lpCtl is a Pointer to an Application-Supported LACTL-Structure control-Structure.

Example:

The following small example Procedure extracts FILE_ID.DIZ from Archiv C:\\LZAPI.LZH into Directory "C:\\TMP".

```
void GetTheDescription(HWND HDialog)
{LACTL  ctl;
 memset(ctl, 0, sizeof(ctl));
 ctl.lStructSize = sizeof(ctl);
 ctl.hwndOwner = HDialog;
 ctl.lpstrArchivName = "C:\\LZAPI.LZH";
 ctl.lpstrPath = "C:\\TMP";
 ctl.lpstrWildcards = "FILE_ID.DIZ";
 ctl.Flags = LAF_SHORTNAMES;

 LAExtract( &ctl );
 }
}
```

# LAAppend Function

The LAAppend-Function adds one or more Files to an Archiv.

LAERR FAR PASCAL LAAppend(LPLACTL lpCtl);

lpCtl is a Pointer to an Application-Supported LACTL-Structure control-Structure.

Example:

The following small example Procedure creates an Archiv with all INI-Files in the Windows-Directory in Archiv C:\TMP.LZH.

```
void GetTheDescription(HWND HDialog)
{LACTL  ctl;
 char   WindowsDirectory[144];
 GetWindowsDirectory(WindowsDirectory, sizeof(WindowsDirectory) );

 memset(ctl, 0, sizeof(ctl));
 ctl.lStructSize = sizeof(ctl);
 ctl.hwndOwner = HDialog;
 ctl.lpstrArchivName = "C:\\TMP.LZH";
 ctl.lpstrPath = WindowsDirectory;
 ctl.lpstrWildcards = "*.INI";
 ctl.Flags = LAF_SHORTNAMES;

 LAAppend( &ctl );
 }
}
```

# LADelete Function

The LADelete-Function removes one or more Files from an Archiv.

LAERR FAR PASCAL LADelete(LPLACTL lpCtl);

lpCtl is a Pointer to an Application-Supported LACTL-Structure control-Structure.

Example:

The following small example Procedure deletes all "BAK"-Files from the Arcgiv C:\TMP.LZH.

```
void DeleteBakFiles(HWND HDialog)
{LACTL  ctl;

 memset(ctl, 0, sizeof(ctl));
 ctl.lStructSize = sizeof(ctl);
 ctl.hwndOwner = HDialog;
 ctl.lpstrArchivName = "C:\\TMP.LZH";
 ctl.lpstrWildcards = "*.BAK";
 ctl.Flags = LAF_SHORTNAMES;

 LADelete( &ctl );
 }
}
```

# Using a Callback-Function

Here are some reasons why you may need to Use an Application-Supplied callback-Function:

- Without a callback-Funktion "Multitasking" stops when Compressing/Decompressing Files.

- You may want to select Files to decompress.
- You may want to find out things about a given Archiv.
- You may want to give the User a chance to cancel the Compression/Decompression Process.

### Declare your Callback

Your Callback should look like this:

For C++:
extern "C" LAERR FAR PASCAL TCatCallBack(int msg, LPLACTL p);

for C:
LAERR FAR PASCAL TCatCallBack(int msg, LPLACTL p);

"m" gives you al LZAPI-Message-Index. Currently the only messages defined are LHM_PEEK and LHM_NEXTFILE.

"p" gives you Information about the LACTL-Control-Structure being used for the Process.

Here is an Example for an LZAPI-Callback-Function

```
// --------- Callback-Funktion for LZAPI-Bibliothek --------
extern "C" LAERR FAR PASCAL _export TCatCallBack(int msg, LPLACTL p)
{ switch (msg) {
     case LAM_PEEK:    GetApplicationObject()->PumpWaitingMessages();
                  if (CancelImmediatly)
                    return LAN_STOP;
                  break;
  }
  return 0;
}
```

LAM_PEEK comes from Time to Time to inform you that you've grown a little bit Older. It's main Purpose is to allow You to Use Windows API PeekMessage/DispatchMessage to cooperate win Windows 3.x-"Multitasking". You may stop a Compression/Decompression Process by returning LHN_STOP, but ir can't be guaranteed that all Files will be closed correctly etc.

Note that during the Processing of the LAM_PEEK-Message the Current Directoy may be changed if the Work is done by a DOS-Box based Archiver.

LAM_NEXTFILE will be called any time the next File is to be Processed during compression and decompression in an archiver-DLL. This Message does not occur when LZAPI has to call a DOS-Based Archiver like RAR.EXE to do the Work.

LAM_PERCENTOFFILE will be called from time to time to give your Application a chance to maintain a Process-Indicator control. The uPercentOfFile-Member of the Control-Structure will contein the percentage of the data currently being compressed or extracted.

time to time to give your Application a chance to maintain a Process-Indicator control. The uPercentOfFile-Member of the Control-Structure will contein the percentage of the data currently being compressed or extracted.

LAM_MAPNAME will be called from WUNZIP.DLL immediatly before it Opens the destination-File for Decompression. The application may Overwrite the Filename in the Buffer pointed to by the `lpstrProcessedFile` Member with the Name required for extraction.

Example:

```
int FAR PASCAL _export ExampleCallback(int msg, LPLACTL Ctl)
{ switch (msg) {
    case LAM_MAPNAME: if (!lstrcmpi(Ctl->lpstrProcessedFile,
"WIN.INI"))
                      lstrcpy (Ctl->lpstrProcessedFile,
"NEWNAME.FIL");
                      break;

  }
  return 0;
}
```

Note that you should use "smart Callbacks" or "MakeProcInstance" for this Callback-Funktion like any other Callback-Funktion Transfered thru DLL-Boundaries.

# UI-Helper Functions

Some User-Interface supporting Functions are included in LZAPI. Their Purpose is to find out, which kind of Files are currently supported. Future versions of LZAPI, which support even more Archiv Types and Packers will add these Informations to your Application with the Help of these Functions

### File-Open Support

When displaying a "File Open" Dialog-Box you may want to give the User a list of available Extensions, for Example "*.LZH;*.ARJ;*.ZIP" and so on.

The Function LAGetExtensionString will give you the string needed for this Purpose.

LAGetExtensionString will search for each unique DLL or DOS-Archiver needed to list an Archiv. For example, if WLHA.DLL is not available, it will search for LHA.EXE, if neither of both is available, it will omit *.LZH from the List of Extensions. So your User will always see the files he can actually Process with his Installation.

### File-Decision Support

Function LAIsArchive will inform you if a given File name is an Archiv supported by LZAPI.

LAIsArchive works in two steps:
1. It will decide the Archiv-Type by itÄs extension (LZH,ZIP,ARJ...)
2. If this does not Help, as for SFX-Archives in EXE-Files, LAIsArchive will open the File and try to analyze its contents.

LAIsArchive returns Zero (TY_NONE), when the Archiv-Type is unknown, it returns a TY_xxxx - Constant when the File has been detected to be an Archiv.

# DLL Teamwork

The whole LZAPI-System consists of a lot of DLLs. This Chapter gives you an introduction in the Teamwork LZAPI performs for you to achieve its Tasks.

### Standard Operation

Normally, your Program will communicate directly with the LZAPI.DLL only. All the other DLLs will be loaded by LZAPI.DLL when needed.

LZAPI has a built-in list of Functions and DLLs. It "knows", which DLLs have been configured for it's use and scans its list to achieve a specified Functionality.

If a DLL can't be found on the PATH, LZAPI will try to run the DOS-Version of a Packer via a DOS-Prompt. That means, if you do not include any DLL but LZAPI.DLL with your Program, it will continue to work, yet a bit slower. If also the DOS-Version of the Packer ist not available, LZAPI will inform the End-User about the Filename of the Packer it needs.

### Finding the DLL

LZAPI will select the DLL depending on the File-Extension of the Archiv and the Function to achieve. If the File Extension gives no Information it will load Data from the File and analyze it to find out about the File Type. It will search for the DLL on the Path and load it via a "LoadLibrary"-Call.

### Finding the Function

Each Wxxxx.DLL exports some Functions of a set of Functions. The Name of the Function includes the Archive Extension:

**16 Bit Edition:**

| Library Name | Function Names |
|---|---|
| WARC.DLL | ARCAppend |
| | ARCDelete |
| | ARCExtract |
| | ARCList |
| WLHA.DLL | LZHAppend |
| | LZHList |
| | LZHExtract |
| WUNARJ.DLL | ARJExtract |
| | ARJList |
| WUNZIP.DLL | ZIPExtract |
| | ZIPList |
| WZIP.DLL | ZIPAppend |
| | ZIPDelete |
| WZOO.DLL | ZOOAppend |
| | ZOOExtract |
| | ZOOList |

**32 Bit Edition:**

| Library Name | Function Names |
|---|---|
| WARC32.DLL | ARCAppend |
| | ARCDelete |

| | |
|---|---|
| | ARCExtract |
| | ARCList |
| WLHA32.DLL | LZHAppend |
| | LZHList |
| | LZHExtract |
| | LZHDelete |
| WUNARJ32.DLL | ARJExtract |
| | ARJList |
| WUNZIP32.DLL | ZIPExtract |
| | ZIPList |
| WZIP32.DLL | ZIPAppend |
| | ZIPDelete |
| WZOO32.DLL | ZOOAppend |
| | ZOOExtract |
| | ZOOList |

All of these Functions have the same Function-Prototype:

LAERR FAR PASCAL FunctionName(LPLACTL);

### Speeding up

LZAPI is an abstraction Layer for your Program to use all of the DLLs listed above without thinking a lot about the Functionality needed. LZAPI will select the DLL needed and load it.

However, Loading a DLL via a Call to LoadLibrary is not one of the fastest things to do.

You can help LZAPI with it's job by loading the DLL at Program startup Time.

Example:
A Program needs to work with a ZIP-File and calls LAAppend 10 Times. LZAPI will 10 Times Call LoadLibrary and afterwards FreeLibrary again, so the overhead of loading the DLL will be needed 10 Times.

You can increase Speed by Loading WZIP.DLL with LoadLibrary at Program Startup and freeing it with FreeLibrary when your Program terminates.

### Skipping LZAPI

If you do not need the Abstraction Technology of LZAPI, you may skip it and work directly with the exported Functions of the Wxxxxx.DLLs.

To do that, simply generate an Import-Lib using the IMPLIB-Utility shipped with your Compiler and include this LIB in your Link-Project (Or declare the function as external in Pascal and Visual Basic).

# Specialities of Archivers

Here are some interesting Informations that will help you to select the Archiver Technology best to your Appliaction.

### WLHA.DLL

WLHA.DLL is a Port of the Sources from Haruyasu Yoshizaki. The greatest part of it has been written in Assembly Language, so it is by far the fastest Archiver/Dearchiver in the LZAPI-Package.

WLHA brings with it it's own Header-File WLHA.H, that shows out a much more detailed API than the normal LZAPI. WLHA supports a techology called MFILE, which is a Stream-Handle that may reside in Memory, so Memory <-> Memory-Compression etc. become Possible.

Because of the many Assembly-Language modules that make WLHA so fast, WLHA.DLL is least likely to be ported to WIN32 soon.

### WARC.DLL

WARC is a Port of the Sources from Dietmar Bückart. These are Pascal-Sources, so Porting of WARC.DLL to 32 Bit depends one Borlands release of a 32-Bit-Pascal Compiler.

ARC-Archives can not contain long Filenames (With Directory names), their structure and Compression is relatively Simple.

### WUNARJ.DLL

WUNARJ is a Port of the Sources by Robert K. Jung. Rober K. Jung did never release the Sources of ARJ, so there does not exist and ARJ-Compression DLL and Compression for ARJ-will not be supported by LZAPI.

ARJ-Compression is supported by the use of a DOS-Box when ARJ.EXE is available on your System.

### WUNZIP.DLL / WZIP.DLL

WUNZIP und WZIP are ports of the quite amazing Sources of the InfoZip Group. They are the most common standard of Archivers.

### TAR

TAR is initially a File-Format that comes from Unix-Machines. It is not the way Archivers like PKZIP are, because it does not support File Compression, only Storing of Files is supported.

Note, that Tar does not support to add files to an existing Archiv. Archives will therefore be created from scratch anytime you want to add a File.

TAR is supported by LZAPI thru the use of a DOS-Box, if you have TAR.EXE installed on your system Path.

### RAR

RAR is one of the newest Archivers currently available. It's benefits are an easy-to-use DOS-Shell, very good Compression Ratio and limited support of other File Formats.

RAR is supported by LZAPI thru the use of a DOS-Box, if you have RAR.EXE installed on your system Path.

.

See Announcements for details on Sources.

# DOS-Box Archivers

When LZAPI can't find a DLL to perform a specific Archiving Task, it will try to run an Archiver using a DOS-Box.

For example if you did not include WLHA.DLL in your package and the user tries to open a LHA-Archiv thus LAList will be called, LZAPI will search for the DOS-Version of LHA.EXE on the Path and executes a DOS-Command to list the Archiv. Your Program will not notice this behaviour, it's just a bit slower and Error-Reactions are weak.

The following Table will show you which Versions of DOS-Archivers we took to test the LZAPI-Functionality:

| Archiver | Version | Function |
|---|---|---|
| LHA.EXE | 2.12 | LZH All Functions |
| ARJ.EXE | 2.39d | ARJ All Functions |
| PKARC.COM | 3.5 | ARC Creation, Deletion |
| PKXARC.COM | 3.5 | ARC Extraction, Listing |
| PKZIP.EXE | 2.04c | ZIP Creation, Deletion |
| PKUNZIP.EXE | 2.04g | ZIP Extraction, Listing |
| RAR.EXE | 1.50.1beta | RAR All Functions |
| TAR.EXE | | GNU Tar (18.10.1990) |
| ZOO.EXE | 2.01 | ZOO Rahul Dhesi (25.8.1988) |

# Visual Basic Function Prototypes

The following lists our Function Prototypes for LZAPI basic Operation with Visual Basic. We are not yet very expierienced with it....

```
' LZAPI Function Declaraion for Visual Basic


' Bernd Herd: LZAPI Definition

' 1. Get Listing of Archives Contents
 Declare Function LAListDirect Lib "LZAPI.DLL" (ByVal hWnd As Integer, ByVal ArchivFile As String,
ByVal Wildcards As
String, ByVal Listing As String, ByVal dwFlags As Long, ByVal CallBack As Long) As Integer

' 2. Extract a File from an Archiv
 Declare Function LAExtractDirect Lib "LZAPI.DLL" (ByVal hWnd As Integer, ByVal ArchivFile As
String, ByVal DestPath as
String, ByVal Wildcards As String, ByVal dwFlags As Long, ByVal CallBack As Long) As Integer

' 3. Append a File to an Archiv
 Declare Function LAAppendDirect Lib "LZAPI.DLL" (ByVal hWnd As Integer, ByVal ArchivFile As
String, ByVal SourcePath as
String, ByVal Wildcards As String, ByVal dwFlags As Long, ByVal CallBack As Long) As Integer

' 4. Delete a File from an Archiv
 Declare Function LADeleteDirect Lib "LZAPI.DLL" (ByVal hWnd As Integer, ByVal ArchivFile As
String, ByVal Wildcards As
String, ByVal dwFlags As Long, ByVal CallBack As Long) As Integer

' Simple Example to Extract a Listing from an Archiv

   Result$ = String$(10000, 0)
   WildCards$ = "*.*"
   FileName = UCase$(DateiForm.TXT_Textfeld.Text)

   Rtn = LAListDirect(hWnd, DateiName$, WildCards$, Result$, 0, 0)
   TXT_KartenKörper.Text = Result$
```

# For the Future


If you have special needs like

- Reporting an Error.
- Adding special Archivers.
- Adding special LZVIEW-Support
- Building LZAPI or LZVIEW-Based Applications.
- Getting a 32-Bit Version.
- Translating LZVIEW in another Language

Please contact us. We have a Fax-Number and EMail-Adresses for FIDO,CompuServe and Internet printed on out Feedback-Formular.

Don't get angry when finding an Error or missing a Feature. Just send us an electronic Mail, possibly we already fixed the Problem....

### English?

As you may think already, Bernd Herd, who write this Help Text, is not a native English speaker. I hope you're able to excuse my Language difficulties. People able to speak German are welcome to do so...

LZVIEW will run in german on an german installed Windows by loading LZVDEU.DLL. The only thing needed to add other languages would be other Language-Resource-DLLs. If you'd like to use LZVIEW in a larger range, we can send you language-dependend Source-Files in English or german, so you'll be able to generate your own translation.

### Spanish?

You may correspond in Spanish with us, but be shure to write in an easily understandable way.

# License Agreement

LZAPI, DAVINCI and LZVIEW are not and have never been free Software. You have the right to test the Shareware-Version of LZAPI for 30 Days, afterwards you'll have to register (buy) it, or delete it.

LZAPI Licensing licensing price is 99 US $ or 150 DM for a redistributable License. This will contain the License to use the LZAPI/WLHA/WUNZIP/WZIP/WARC/WUNARJ/WZOO - DLLs in your own application and distribute the DLL-files with it.

99 US $ 150 DM 16-Bit Edition only
99 US $ 150 DM 32-Bit Edition only
139 US $ 190 DM 16 and 32-Bit Edition

LZVIEW Archiv-Viewer executable License will cost
15 US $ or 20 DM for ONE License
15 US $ / 20 DM + 2,5 US $ or 3 DM for every additional License up to 50 Licenses
120 US $ / 160 DM + 0,5 US $ or 0,70 DM for every additional License up to 500 Licenses
250 US $ / 350 DM + 0,1 US $ or 0,14 DM for every additional License wanted (Shareware-CD-ROM-Providers special Price). LZVIEW is also available as a 32-Bit Program.

LZVIEW Source-Code Licensing will give you the right to build your own Applications based on the LZVIEW-Source code (The newly designed Application must have a different Name). This License is available for the Price of 250 US $ /350 DM in addition to the LZAPI-License.

DAVINCI The DAVINCI Graphics import Library is available as an own Shareware-Product with its own Documentation for a fee of 69 US $ / 99 DM for a redistributable 16-Bit Version. It is also Available as a 32-Bit-Version.

Please order with the Ordering Form or use CompuServe SWREG.

### Giving you a License

After paying the registration fee Dipl Ing. Bernd Herd give you the right to use the LZAPI Appliaction development Files on one Computer. Using a Software is analog to using a Book: You may use it on one single Computer Simultaniously. Copys are to be done for Backup purposes only.

There are two Editions of LZAPI: 16-Bit Editions and 32-Bit Edition. Both are separate Products and we need to know which Edition you want to register.

You've got the right to sell your Programs with the licensed File-Versions of LZAPI:

| | 16-Bit Edition | 32-Bit Edition |
|---|---|---|
| Basic Support | LZAPI.DLL | LZAPI32.DLL |
| LZH-Support | WLHA.DLL | WLHA32.DLL |
| ZIP-Support | WZIP.DLL | WZIP32.DLL |
| | WUNZIP.DLL | WUNZIP32.DLL |
| ARJ-Support | WUNARJ.DLL | WUNARJ32.DLL |
| ARC-Support | WARC.DLL | WARC32.DLL |
| ZOO-Support | WZOO.DLL | WZOO32.DLL |

You may also distribute Programs linked with LZRES.OBJ or LZRES32.OBJ. Your Programs must

contain a Copyright notice (That means you can't deliver a public domain Program with LZAPI).

You are not allowed to change, disassemble or otherwise reenginere LZAPI or its Component Librarys or Help-System.

### Update Support

LZAPI contains additional Functionality for Registered Users to transfer a Password to LZAPI so any Shareware-Version will work as a registered Full versions. This makes shure you will never have any Problem getting the newest Updates.

### Limited Warranty

We give no Warranty for LZAPI to fit any given purpose. Any Program may be errornous, so we give no warranty for Costs you may have thru using LZAPI.

### License Agreement for ShareWare-Version

The Shareware-Version of LZAPI may be tested for 30 Days.

You also have the right to

- Distribute the unchanged Archiv.
- Test LZAPI on ONE Computer for a maximum of 30 Days.

### Contacting the Developers:

Herd Software Entwicklung
Dipl. Ing. Bernd Herd
Rudolf-Virchow-Str. 8
68642 Bürstadt

Tel.: +49-6206-707775
Fax.: +49-6206-707776

EMail-Adressen:
InterNet: **HerdSoft@aol.com**
CompuServe: 100545,3001
AOL: HerdSoft
FIDO-Netzadresse: 2:2464/420.10.
Word wide Web: http://members.aol.com/herdsoft/index.htm

## Print your Order: 

## or use CompuServe SWREG

# LZRES.OBJ: Handling of Archives stored in your Programs Resources

LZRES.OBJ is an additional Object-File that manages decompression of Archives that are stored within your Applications Resources.

You can use LZRES with Pascal and C/C++ - Programs WITHOUT using a DLL.

### Using LZRES with Borland Pascal 7.0

To use LZRES in Borland Pascal include the unit "LZRES" in your "Uses" Statement.

### Using LZRES with C

Link LZRES.OBJ in your Application and use the LZRES.H - Header-File

### Using the 32-Bit Version of LZRES

The 32-Bit Version of LZRES is available as LZRES32.OBJ for Borland C compiler and LZRES32V.OBJ for Microsoft Visual C++.

### Additional Support for C++ and OWL 2.x

When Programming with Borland C++ you may want to use our TDib and TBitmap-Class extenstion. It consists of the Classes TLZDib and TLZBitmaps. They are upward-Compatible to TDib and TBitmap, but they add additional Constructors to load the Dib or Bitmap from a Resource.

Include LZRES.CPP in your Project and LZRES.H in your Source.

### Putting Archives in your Programs Resources

That's really straightforward.

For example you have a Sign-On-Logo Bitmap-File called "LOGO.BMP" that shall be compressed because it's a bit bigger... First we have to create a compressed Version of it.

LHA -h0 a LOGO LOGO.BMP

Until now you had a BITMAP-statment in your Resource, for Example.

ID_LOGO BITMAP "LOGO.BMP"

Simply replace it with

ID_LOGO ARCHIV "LOGO.LZH"

The bitmap should not be an OS/2-Type Bitmap.

That's all, the Bitmap is compressed in your Resource, now letÄs take a look at getting it out of the Ressource.

### Loading a Bitmap from a Resource

Instead of the Windows API-Call "LoadBitmap" now simply use a Call to "LoadLZHBitmap" and add a Parameter which you set to NULL.

### Loading a DIB from a Resource

If you have to deal with DIBs, Metafiles or other Handles loaded out of your Resource, you normally deal with FindResource and LoadResource.

```
hFind = FindResozrce(hInst, "NAME", "TYPE");
hMem = LoadResource(hInst, hFind);
pMem = LockResource(hMem);

...

UnlockResource(hMem);
FreeResource(hMem),
```

To load a LHA-Compressed Resource just use "LoadLZHResource". But note, that this Function will not return a Resource-Handle but a Global memory-Handle. That means you should NOT use LockResource, UnlockResource and FreeResource. Use GlobalLock, GlobalUnlock and GlobalFree instead.

```
hMem = LoadLZHResource(hInst, "NAME", 0);
pMem = GlobalLock(hMem);

...

GlobalUnlock(hMem);
GlobalFree(hMem),
```

### Omitting Header Data during load

Many File-Formats contain additional File-Headers to the Data represented for it in Memory. For Example Bitmap-Files are leaded by a BITMAPFILEHEADER, that is not included in Standard CF_DIB-Data format.

With the last Parameter of LoadLZHResource you can request to omit the given count of Bytes from the new Global memory handle, thus omitting the BITMAPFILEHEADER.

# Announcements:

WLHA contained in this Software Product is Based on LHA, a Freeware Packer / unpacker Copyright (c) Haruyasu Yoshizaki, 1988-91.

Yoshi allowed the selling of our Setup as a sharware Product.

WLHA.DLL is a Portation of the initially by Yoshi publicated Sources of LHA. Bernd Herd ported the DOS-Sources for a Windows DLL in December-January 1993. The DLL is available as Shareware Product.

WUNARJ is a Port of UNARJ initially released by Robert K. Jung.

WUNZIP and WZIP are Ports of the Sources released by the InfoZip-Group and they initial Sources are copyrighted 1990-1993 Mark Adler, Richard B. Wales, Jean-loup Gailly, Kai Uwe Rommel and Igor Mandrichenko. Contactable at zip-bugs@wkuvx1.bitnet.

WARC is a Port of the Sources released by Dietmar Bückart as CHIPARC.

WZOO is a Port of the Sources released by Rahul Dhesi.

DAVINCI is a Shareware Bitmap Format Import Filter developed by Bernd Herd and available as Shwareware.

LZVIEW has been developed using Borland C++ Version 4.5. It is based on an AppExpert generated Program.

# Paying via CompuServe SWREG

Instead of normal ordering of your License with a Fax or EMail, CompuServe users can also use SWREG to order and buy LZAPI. No additional Charges will be needed. CompuServe will get the Money from you via your normal CompuServe Bill.

To order via SWREG, you will "**GO SWREG**" in Compuserve and follow the instructions there.

LZAPI has SWREG ID-Code **5578**

**EU-Clients please include your international VAT-Number.**

**Print your Order:** 

Herd Software Entwicklung
Bernd Herd
Rudolf-Virchow-Str. 8

**68642 Bürstadt**
Germany

Fax.: 0049-6206-707776

Datum

Tel. 0 62 06 / 707775 FIDO-Netzadresse 2:2464/99.10.
Fax 0 62 06 / 707776
CompuServe: 100545,3001 EMail: herdsoft@aol.com

**Order**

Please send us the following License(s) like stated in the License-Agreement:

| Count | License | Price per DM | Sum DM |
|---|---|---|---|
| ____ | LZAPI common Archiver API | 149,-- | ____ |
| ____ | LZVIEW Archive-Viewer (See Licensing Terms for Prices) | ____ | ____ |
| | Shipping and mail | | 7,-- |
| | | Sum | ____ |

Prices including VAT in DM.

Firma: _____ Umsatzsteuer Ident-Nr: _____

Name: _____ Telephone: _____

Street:_____ Telefax: _____

ZIP/City: _____ EMail/FIDO: _____

Sign:_____

Herd Software Entwicklung
Dipl. Ing. Bernd Herd
Rudolf-Virchow-Str. 8

**68642 Bürstadt**

Tel.: +49-6206-707775
Fax.: +49-6206-707776

EMail-Adressen:
InterNet: HerdSoft@aol.com
CompuServe: 100545,3001
AOL: HerdSoft
FIDO-Netzadresse: 2:2464/420.10.
Word wide Web: http://members.aol.com/herdsoft/index.htm

Datum

## Feedback-Formular

Bitte füllen Sie dieses Formular aus, so daß wir Ihre Wünsche in zukünftigen Entwicklungen berücksichtigen können.

| Für welches Betriebssystem entwickeln Sie hauptsächlich ? | Welche Entwicklungsumgebung benutzen Sie vorzugsweise ? | Welche Art von Entwicklung machen Sie ? |
|---|---|---|
| ☐ Windows 3.1<br>☐ Windows NT<br>☐ OS/2<br>☐ Linux<br>☐ Unix o. ä.<br>☐ MS-DOS<br>☐ Andere<br>_____ | ☐ Borland C++ 3.1<br>☐ Borland C++ 4.x<br>☐ Visual C++<br>☐ Microsoft C<br>☐ Visual Basic<br>☐ Access<br>☐ Paradox<br>☐ Andere<br>_____ | ☐ Anwendungen<br>☐ Anbindungsroutinen<br>☐ Libraries<br>☐ Treiber<br>☐ Schnittstellen<br>☐ Utility<br>☐ andere<br>_____ |

| Wie fanden sie die Dokumentation zu unserem Produkt | Hatten Sie Probleme mit Programmfehlern | Sind Sie mit unserer Preispolitik einverstanden |
|---|---|---|
| ☐ Gut<br>☐ Ausreichend<br>☐ Mangelhaft | ☐ Keine<br>☐ Kaum<br>☐ Unzumutbar viel | ☐ Ja<br>☐ Nein, zu Teuer<br>_____ |
| **Wie fanden sie die Beispiele zu unserem Produkt** | **Sind sie an weiteren Produkten von uns interessiert** | |
| ☐ Gut<br>☐ Ausreichend<br>☐ Nicht | ☐ Ja<br>☐ Nein | |

Was wünschen Sie sich für die zukünftige Weiterentwicklung

_____

_____

Was hast Sie an der vorliegenden Version am Meisten gestört

_____

_____

Name: _____

Straße:_____

PLZ/Ort: _____

Telefon/Fax:_____

EMail/FIDO...: _____

# *Direct WLHA Usage:* Using the extended Interface directly

# WLHA.DLL

*LHA Compression / Decompression Library*

## Using WLHA.DLL without LZAPI.DLL

Like any of the Other Wxxx.DLLs included with LZAPI you can use WLHA.DLL without the LZAPI abstraction layer. Just link to WLHA.LIB instead of LZAPI.LIB and use calls to LZHAppend instead of LAAppend, LZHExtract instead of LAExtract and LZHList instead of LAList. So you can use LZH-Archives with one DLL only and you do not have to copy two or more DLLs.

## The extended WLHA Interface

But WLHA.DLL exports some additional Functions to support detailed Control, Streaming-Technology, in-memory-Compression etc.

To use it include WLHA.H or "uses WLHA.PAS" in your Application.

## MFILEs

All the File I/O Functions of WLHA are based on MFILE-Handles. These Handles are much like the stream-Based File I/O-Functions of C/C++ - Programs : fopen/fclose/fread etc. But they are defined in a more abstract way so they can be combined to perform I/O to Windows Global memory Handles etc.

## Basic usage

**Step 1:**

include WLHA.H in your Source File

**Step 2:**

Run IMPLIB and include WLHA.LIB in your Project(link) File.

**Step 3:**

Because WLHA can only handle one Client, you have to ask for the WLHA-Usage before beginning of Compression / Decompression Events.

Therfore you need to Call LHInit, giving it the Instance-Handle of Your Application. This is only Important of your Program should be "multitasking"-Capable.

**Step 4:**

For File Compression purposes you use the LHAppend-Function.

LHAppend creates, overwerits or appends to an LHA-Archiv depending on the Options-Parameter:

You should set the LHA_CREATEARCHIV in the Options-Parameter to create a new Archiv or overwrite an existing one.

We also recommend to use LHA_SHORTNAMES.

**Step 5:**

Call LHEnd with your Instance-Handle to end the WLHA-Usage.

This results in the following Code example:

```
LHERR e;                    // Error result Variable
extern HINSTANCE hInst;

if (      LHE_OK != (e=LHInit(hInst) ) ||
          LHE_OK != (e=LHAppend("MYARC.LZH", "FIRSTFIL.TXT", LHA_CREATEARCHIV |
LHA_SHORTNAMES) ) ||
          LHE_OK != (e=LHAppend("MYARC.LZH", "NEXTFIL.TXT", LHA_SHORTNAMES) ) ||
          LHE_OK != (e=LHEnd(hInst) ) ) )
          {
              LHErrMsgBox(e);
              LHEnd(hInst);
          }
```

Please Use the ">>"-Button to go to the next Help Page.

# *Direct WLHA Usage:* Using a Callback-Function

Here are some reasons why you may need to Use an Application-Supplied callback-Function:

- Without a callback-Funktion "Multitasking" stops when Compressing/Decompressing Files.

- You may want to select Files to decompress.
- You may want to find out things about a given Archiv.
- You may want to give the User a chance to cancel the Compression/Decompression Process.

## Declare your Callback

Your Callback should look like this:

For C++:
extern "C" LHERR FAR PASCAL TCatCallBack(int msg, LPLHHEAD p);

for C:
LHERR FAR PASCAL TCatCallBack(int m, LPLHHEAD p);

"m" gives you al LHA-Mesage-Index. Currently the only messages defined are LHM_PEEK and LHM_NEXTFILE.

"p" gives you Information about the LHA-Archive currently being processed.

Here is an Example for an WLHA-Callback-Function

```
// --------- Callback-Funktion für LHA-Bibliothek --------
extern "C" LHERR FAR PASCAL _export TCatCallBack(int msg, LPLHHEAD p)
{ switch (msg) {
      case LHM_NEXTFILE:    fprintf(outfile, "%s %ld", p->filename, p->original);
                            return 0;
  }

  return LHDefCallbck(msg, p);
}
```

LHM_PEEK comes from Time to Time to inform you that you've grown a little bit Older. It's main Purpose is to allow You to Use Windows API PeekMessage/DispatchMessage to cooperate win Windows 3.x-"Multitasking". You may stop a Compression/Decompression Process by returning LHN_STOP.

LHM_NEXTFILE will be called any time the next File is to be Processed during decompression. You may want to create a list of the Archives contents or select some Files for Decompression only. Return 0, if you don't want the File to be processed (Decompressed), return LHN_STOP, if decompression shall end immediatly, or call LHDefCallback to return the correct value for normal Decompression (a MFILE-Handle).

Note that you should use "smart Callbacks" or "MakeProcInstance" for this Callback-Funktion like any other Callback-Funktion Transfered thru DLL-Boundaries.

## Transfer the Callback-Procedure Address

Call LHSetCallback to Inform WLHA about the Addess of the Callback-Procedure that shall be used for the next operations. Note that You have to Transfer the Callback-Proecdure-Adress again after Calling LHEnd/LHInit again.

Don't forget to use LHSetCallback(NULL) when your Callback is not prepared for being executed any more.

## ***Direct WLHA Usage:*** **Using Memory Data Blocks**

The WLHA-High-Level routines LHExpand and LHAppend are defined to use the mopen-Routine to come to their MFILE-Handle.

### LAAppendNext

```
LHERR LHF LHAppendNext(
          MFILE  marc,       // File Handle Archiv
          MFILE  minp,       // File Handle Input File
          LPCSTR FileName,   // FileName Inside Archiv
          int    Options);   // Archiving Options
```

LAAppendNext allows you to Append a MFILE opened with mopen or mopenglobal to another MFILE-Handle.

Example:

```
HGLOBAL CompressHandle(HGLOBAL hGlobalHandle)
{ MFILE input = mopenglobal(hGlobalHandle,
GlobalSize(hGlobalHandle) );
  MFILE output = mopen("OUTFILE.LZH", OF_CREATE | OF_READWRITE);

  LHAppendNext(output,
     input,
     "NAME.EXT",
     0);

  mclose(input);
  return hGlobalHandle = (HGLOBAL) mclose(output);
}
```

### LAExpandNext

```
LHERR LHF LHExpandNext(
          MFILE  marc,       // File Handle Archiv
          LPCSTR Wildcards,  // Wildcards (e.g. *.TXT) or NULL
          int    Options);   // Decompression Options
```

LAExpandNext extracts all the Files from the Archiv given my the MFILE-Handle. To get MFILE-Handle for the Destionation Files, LHExpandNext calls the CallBack-Funktion given by LASetCallback with the LAM_NEXTFILE-Message. The Callback-Function should return a MFILE-Handle.

# Other Products from Herd Software Development

You may want to take a look onto other shareware distributed programs available from our team:

| Compuserve FileName | Compuserve Forum | Description | Price | Document-Language |
|---|---|---|---|---|
| LZAPI.ZIP | WINSDK | Your Doorway to the Archive: A set of redistributable DLLs giving your applications access to compressed archives in formats ZIP,ARC,LZH,ARJ,RAR,ZOO and TAR | 99 US $ | English |
| JPGANI.ZIP | WINSDK | Available for 16-Bit and 32-Bit Windows development LZH- and JPEG compressed pictures in WinHelp (.HLP) and Media Viewer (.MVB) files. Allows for photographic-like use of WinHelp multimedia Applications. | 20 US $ | English |
| ASETUP.LZH | GERWIN | Multilungual german /english/french installation program with C-source, fast LZH decompressor, self-installing EXE capability, easy to use | Public Domain | English + German |
| HEXE.LZH | GERWIN | German edition of the ASetup Drag & Drop Workshop | 200 DM | German |
| WITCH.LZH | WINSDK | English edition of the ASetup Drag & Drop Workshop | 150 US $ | English |
| HLP2RTF.ZIP | PCPROG | Help file to RTF converter<br>- Allows to Converrt a help file or Media Veiwer file into a recompilable HPJ project<br>- or to Convert it for printing including all the bitmaps | 40 US $ | English |
| NEUBOX.LZH | GERWIN | Custom Controls for database applications.<br>- Column listbox with user editable fields, sizeable headings, sort etc., fast filling up to 32000 lines<br>- columned ComboBox<br>- Edit control with overwrite mode<br>and many many more... | 499 DM | German |
| GRBTN.LZH | GERWIN | Custom Controls to design dialog boxes<br>- Easy to use Tabbed Dialogs. Allows to use Tabbed Dialogs changing only the resources without changing the source codes of your program.<br>- Easy to use picture buttons<br>- Statusbar automization for dialog boxes<br>and many many more... | 150 DM | German |
| DAVINCI.LZH | GERWIN | Graphic Import/Export library. Imports graphics as device independend bitmaps and metafiles:<br>- Import: BMP/WMF/TIF/JPG/PCX/DXF/GIF<br>- Export BMP/WMF/TIF/JPG<br>- Transparently allows to use Aldus / Microsoft import filter programs (.FLT)<br>- Includes powerfull set of graphic support gunctions like dithering, rotating... | 150 DM | German |
| LHALIB14.ZIP | WINSDK | 3,5 KByte LHA decompression DLL for setup applications with assembly language source | Public Domain | English |

You might also like to visit our WWW-Pages at

http://ourworld.compuserve.com/homepages/herd_nuding

Most tools listet above can be downloaded from these pages.