Microsoft Cabarc User's Guide

Abstract

This document is the user's guide for the Microsoft program CABARC.EXE, which can create, view, and extract from Microsoft cabinet files.

Table of Contents

Introduction

The cabinet format

The cabinet format provides a way to efficiently package multiple files. The key features of the cabinet format are that multiple files may be stored in a single cabinet ("CAB file"); and that data compression is performed across file boundaries, significantly improving the compression ratio.

Depending upon the number of files to be compressed, and the expected access patterns (sequential or random access; whether most of the files will be requested at once or only a small portion), cabinets can be constructed in different ways. One key concept of the cabinet file is the *folder*. A *folder* is a collection of one or more files which are compressed together as a single entity. By compressing files in this way, the compression ratio is improved. The downside is that random access time suffers, since in order for any particular file in a folder to be decoded, all preceding files in the same folder must also be decoded.

Cabarc

Cabarc is a utility that creates, extracts, and lists the contents of cabinet files (CABs), using a command line interface similar to that of popular archiving tools. Cabarc supports wildcards and recursive directory searches.

Command line usage

Cabarc is used as follows:

```
Usage: CABARC [options] command cabfile [@list] [files] [dest_dir]
```

Currently, only three commands are supported; N (create new cabinet), L (list contents of an existing cabinet), and X (extract files from a cabinet). These commands are described in the following pages.

Options must appear before the command name, and cannot be combined (for example, to set the –r and –p options, use –r –p, and not –rp).

Creating cabinets

Cabinets are created using the *n* command, followed by the name of the cabinet to create, followed by a filename list, as shown below:

```
cabarc n mycab.cab prog.c prog.h prog.exe readme.txt
```

The above command creates the cabinet *mycab.cab* containing the files “prog.c”, “prog.h”, “prog.exe”, and “readme.txt”, in a single folder, using the default compression mode, MSZIP.

Wildcards

Cabarc supports wildcards in the filename list, as shown in the example below:

```
cabarc n mycab.cab prog.* readme.txt
```

Folders

By default, all files are added to a single folder (compression history) in the cabinet. It is possible to tell cabarc to begin a new folder, by inserting the plus (+) symbol as a file to be added, as shown below:

```
cabarc n mycab.cab test.c main.c + test.exe *.obj
```

The above command creates the cabinet “mycab.cab” with one folder containing “test.c” and “main.c”, and a second folder containing “test.exe” and all files matching “*.obj”.

Path name preservation

By default, directory names are not preserved in the cabinet; only the filename component is stored. For example, the following command will result in the filename “prog.c” being stored in the cabinet:

```
cabarc n mycab.cab c:\source\myproj\prog.c
```

In order to preserve path names, the –p option should be used:

```
cabarc –p n mycab.cab c:\mysource\myproj\prog.c
```

This command will cause the file to be named “mysource\myproj\prog.c” in the cabinet. Note that the c:\ prefix is still stripped from the filename; cabarc will not allow absolute paths to be stored in the cabinet, nor will it extract such absolute paths.

Path stripping

In many situations it may be desirable to preserve some of the path name, but not all of it. For example, one might wish to archive everything in the c:\mysource\myproj\ directory, but store only the myproj\ component of the path. This can be accomplished with the path stripping option, -P (capital P).

```
cabarc –p –P mysource\ n mycab.cab c:\mysource\myproj\prog.c
```

The –P option strips any strings which begin with the provided string (wildcards are not supported in this case; it is a simple text match). Any absolute path prefixes such as c:\ or \ are stripped before the comparison takes place, so these characters should not be included in the –P option.

The –P option may be used multiple times to strip out multiple paths; cabarc builds a list of all paths to be stripped, and applies only the first one which matches. For example:

```
cabarc –p –P mysrc\ –P yoursrc\ n mycab.cab c:\mysrc\myproj\*.*
d:\yoursrc\yourproj\*.c
```

The trailing slash at the end of the path name is important; entering `–P mysrc` instead of `–P mysrc\` would cause files to be added as "\myproj\<filename>".

Recursive directory search

Cabarc can archive files in a directory and all of its subdirectories, by use of the –r option. For example, the command shown below will archive all files ending in .h which are in c:\msdev\include\, c:\msdev\include\sys, and c:\msdev\include\gl (assuming these directories exist on your system).

```
cabarc –r –p n mycab.cab c:\msdev\include\*.h
```

The –p option is used here to preserve the path information when the files are added to the cabinet; without this option, only the filename components would be stored, although sometimes it might be desirable behavior to not use –p.

Reserve space for code signature

Cabarc can reserve space in the cabinet for a code signature. This is done using the –s option, which reserves a specified amount of empty space in the cabinet. For code signing, 6144 bytes need to be reserved:

```
cabarc –s 6144 n mycab.cab test.exe
```

Note that the –s option does not actually write the code signature; it merely reserves space for it in the cabinet. The appropriate code signing utility must be used to fill out the code signature.

Set cabinet ID

Cabinet files have a 16-bit cabinet ID field that is designed for application use. The default value of this field is zero, however, the –i option of cabarc can be used to set this field to any 16-bit value:

```
cabarc –i 12345 n mycab.cab test.exe
```

Set compression type

The default compression type for a cabinet is MSZIP. However, the compression type can be changed with the –m option. Currently only MSZIP compression (-m MSZIP) and no compression (-m NONE) are supported.

The following command stores files in the cabinet with no compression:

```
cabarc –m NONE n mycab.c *.*
```

File list from a File

Cabarc can input its list of files from a text file, instead of from the command line, by using @files (“at files”). This is done by prefixing with the @ symbol the name of the file which contains the file list. For example:

```
cabarc n mycab.cab @filelist.txt
```

The text file must list the physical file names of the files to be added, one per line. As is the case when specifying filenames on the command line, the plus (+) symbol can be used as a filename to specify the beginning of a new folder. If a filename contains any embedded spaces, it must be enclosed as quotes, as shown below:

```
test.c
myapp.exe
“output file.exe”
```

The reason for requiring quotes is that each physical filename may be followed on the same line by an optional logical filename, which specifies the name under which the file will be stored in the cabinet:

```
test.c myapp.c
myapp.exe
“output file.exe” foobar.exe
```

If the logical filename contains spaces, then it must also be enclosed in quotes. Note that the logical filename overrides the –p (preserve path names) and –P (strip path name) options -the file will be added to the cabinet *exactly* as indicated. Wildcards are allowed in the physical filename, but in this situation a logical filename is not allowed.

The “@” feature may be used multiple times, to retrieve file lists from multiple files. Cabarc does not check for the presence of duplicate files, so if the same physical file appears in multiple file lists, it will be added to the cabinet multiple times.

The “@” feature may be combined with filenames on the command line. Files are added in the order in which they are parsed on the command line. Example:

```
cabarc n mycab.cab @filelist1.txt *.c @filelist2.txt *.h
```

Note: The “@” feature is available only when creating cabinets, not when extracting or listing cabinets.

List cabinet contents

It is possible to view the contents of a cabinet using the L (list) command, as shown below:

```
cabarc l mycab.cab
```

Cabarc will display the Set ID in the cabinet (see the –s option for cabinet creation), as well as the name of each file in the cabinet, along with its file size, file date, file time, and file attributes.

Extracting cabinets

The X (extract) command extracts files from a cabinet. The simplest use of the X command is shown below, which causes all files to be extracted from the cabinet:

```
cabarc x mycab.cab
```

Alternatively, it is possible to selectively extract files, by providing a list of filenames and/or wildcards:

```
cabarc x mycab.cab readme.txt *.exe *.c
```

By default, full path names (if they are present in the cabinet) are *not* preserved upon extraction. For example, if a file named *mysrc\myproj\test.c* is present in the cabinet, then the command `cabarc x mycab.cab` will cause the file *test.c* to be extracted into the current directory. In order to preserve file names upon extraction, the –p option must be used. This option will cause any required directories to be created if necessary.

Only the filename component is considered in the matching process; the pathname is discounted. For example, `cabarc x mycab.cab test.c` will cause the file *mysrc\myproj\test.c* to be extracted to the current directory as *test.c*, as will `cabarc x mycab.cab *.c` (which will also extract any other files matching *.c).

By default, the extracted files are stored in the current directory (and its subdirectories, if –p is used). However, it is possible to specify a destination directory for the extracted files. This is accomplished by appending a directory name to the command line. The directory name must end in a backslash (\ ). Examples:

```
cabarc x mycab.cab c:\somedir\

cabarc x mycab.cab *.exe c:\somedir\
```