NAME

Archive::Zip::FAQ − Answers to a few frequently asked questions about Archive::Zip

DESCRIPTION

It seems that I keep answering the same questions over and over again. I assume that this is because my documentation is deficient, rather than that people don't read the documentation.

So this FAQ is an attempt to cut down on the number of personal answers I have to give. At least I can now say "You *did* read the FAQ, right?".

The questions are not in any particular order. The answers assume the current version of Archive::Zip; some of the answers depend on newly added/fixed functionality.

Install problems on RedHat 8 or 9 with Perl 5.8.0

Q: Archive::Zip won't install on my RedHat 9 system! It's broke!

A: This has become something of a FAQ. Basically, RedHat broke some versions of Perl by setting LANG to UTF8. They apparently have a fixed version out as an update.

You might try running CPAN or creating your Makefile after exporting the LANG environment variable as

`LANG=C`

<https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=87682>

Why is my zip file so big?

Q: My zip file is actually bigger than what I stored in it! Why?

A: Some things to make sure of:

Make sure that you are requesting COMPRESSION_DEFLATED if you are storing strings.
: `$member−>`desiredCompressionMethod(COMPRESSION_DEFLATED);

Don't make lots of little files if you can help it.
: Since zip computes the compression tables for each member, small members without much entropy won't compress well. Instead, if you've got lots of repeated strings in your data, try to combine them into one big member.

Make sure that you are requesting COMPRESSION_STORED if you are storing things that are already compressed.
: If you're storing a .zip, .jpg, .mp3, or other compressed file in a zip, then don't compress them again. They'll get bigger.

Sample code?

Q: Can you send me code to do (whatever)?

A: Have you looked in the `examples/` directory yet? It contains:

examples/calcSizes.pl	— How to find out how big a Zip file will be before writing it
examples/copy.pl	— Copies one Zip file to another
examples/extract.pl	— extract file(s) from a Zip
examples/mailZip.pl	— make and mail a zip file
examples/mfh.pl	— demo for use of MockFileHandle
examples/readScalar.pl	— shows how to use IO::Scalar as the source of a Zip read
examples/selfex.pl	— a brief example of a self-extracting Zip
examples/unzipAll.pl	— uses Archive::Zip::Tree to unzip an entire Zip
examples/updateZip.pl	— shows how to read/modify/write a Zip
examples/updateTree.pl	— shows how to update a Zip in place
examples/writeScalar.pl	— shows how to use IO::Scalar as the destination of a Zip write
examples/writeScalar2.pl	— shows how to use IO::String as the destination of a Zip write
examples/zip.pl	— Constructs a Zip file
examples/zipcheck.pl	— One way to check a Zip file for validity

examples/zipinfo.pl	— Prints out information about a Zip archive file
examples/zipGrep.pl	— Searches for text in Zip files
examples/ziptest.pl	— Lists a Zip file and checks member CRCs
examples/ziprecent.pl	— Puts recent files into a zipfile
examples/ziptest.pl	— Another way to check a Zip file for validity

Can't Read/modify/write same Zip file

Q: Why can't I open a Zip file, add a member, and write it back? I get an error message when I try.

A: Because Archive::Zip doesn't (and can't, generally) read file contents into memory, the original Zip file is required to stay around until the writing of the new file is completed.

The best way to do this is to write the Zip to a temporary file and then rename the temporary file to have the old name (possibly after deleting the old one).

Archive::Zip v1.02 added the archive methods `overwrite()` and `overwriteAs()` to do this simply and carefully.

See `examples/updateZip.pl` for an example of this technique.

File creation time not set

Q: Upon extracting files, I see that their modification (and access) times are set to the time in the Zip archive. However, their creation time is not set to the same time. Why?

A: Mostly because Perl doesn't give cross-platform access to *creation time*. Indeed, many systems (like Unix) don't support such a concept. However, if yours does, you can easily set it. Get the modification time from the member using `lastModTime()`.

Can't use Archive::Zip on gzip files

Q: Can I use Archive::Zip to extract Unix gzip files?

A: No.

There is a distinction between Unix gzip files, and Zip archives that also can use the gzip compression.

Depending on the format of the gzip file, you can use Compress::Raw::Zlib, or Archive::Tar to decompress it (and de-archive it in the case of Tar files).

You can unzip PKZIP/WinZip/etc/ archives using Archive::Zip (that's what it's for) as long as any compressed members are compressed using Deflate compression.

Add a directory/tree to a Zip

Q: How can I add a directory (or tree) full of files to a Zip?

A: You can use the Archive::Zip::addTree*() methods:

```
use Archive::Zip;
my $zip = Archive::Zip->new();
# add all readable files and directories below . as xyz/*
$zip->addTree( '.', 'xyz' );
# add all readable plain files below /abc as def/*
$zip->addTree( '/abc', 'def', sub { -f && -r } );
# add all .c files below /tmp as stuff/*
$zip->addTreeMatching( '/tmp', 'stuff', '\.c$' );
# add all .o files below /tmp as stuff/* if they aren't writable
$zip->addTreeMatching( '/tmp', 'stuff', '\.o$', sub { ! -w } );
# add all .so files below /tmp that are smaller than 200 bytes as stuff/*
$zip->addTreeMatching( '/tmp', 'stuff', '\.o$', sub { -s < 200 } );
# and write them into a file
$zip->writeToFileNamed('xxx.zip');
```

Extract a directory/tree

Q: How can I extract some (or all) files from a Zip into a different directory?

A: You can use the **Archive::Zip::extractTree()** method: ??? ||

```
        # now extract the same files into /tmpx
        $zip->extractTree( 'stuff', '/tmpx' );
```

Update a directory/tree

Q: How can I update a Zip from a directory tree, adding or replacing only the newer files?

A: You can use the **Archive::Zip::updateTree()** method that was added in version 1.09.

Zip times might be off by 1 second

Q: It bothers me greatly that my file times are wrong by one second about half the time. Why don't you do something about it?

A: Get over it. This is a result of the Zip format storing times in DOS format, which has a resolution of only two seconds.

Zip times don't include time zone information

Q: My file times don't respect time zones. What gives?

A: If this is important to you, please submit patches to read the various Extra Fields that encode times with time zones. I'm just using the DOS Date/Time, which doesn't have a time zone.

How do I make a self-extracting Zip

Q: I want to make a self-extracting Zip file. Can I do this?

A: Yes. You can write a self-extracting archive stub (that is, a version of unzip) to the output filehandle that you pass to **writeToFileHandle()**. See examples/selfex.pl for how to write a self-extracting archive.

However, you should understand that this will only work on one kind of platform (the one for which the stub was compiled).

How can I deal with Zips with prepended garbage (i.e. from Sircam)

Q: How can I tell if a Zip has been damaged by adding garbage to the beginning or inside the file?

A: I added code for this for the Amavis virus scanner. You can query archives for their 'eocdOffset' property, which should be 0:

```
  if ($zip->eocdOffset > 0)
    { warn($zip->eocdOffset . " bytes of garbage at beginning or within Zip") }
```

When members are extracted, this offset will be used to adjust the start of the member if necessary.

Can't extract Shrunk files

Q: I'm trying to extract a file out of a Zip produced by PKZIP, and keep getting this error message:

```
  error: Unsupported compression combination: read 6, write 0
```

A: You can't uncompress this archive member. Archive::Zip only supports uncompressed members, and compressed members that are compressed using the compression supported by Compress::Raw::Zlib. That means only Deflated and Stored members.

Your file is compressed using the Shrink format, which is not supported by Compress::Raw::Zlib.

You could, perhaps, use a command-line UnZip program (like the Info-Zip one) to extract this.

Can't do decryption

Q: How do I decrypt encrypted Zip members?

A: With some other program or library. Archive::Zip doesn't support decryption, and probably never will (unless *you* write it).

How to test file integrity?

Q: How can Archive::Zip can test the validity of a Zip file?

A: If you try to decompress the file, the gzip streams will report errors if you have garbage. Most of the time.

If you try to open the file and a central directory structure can't be found, an error will be reported.

When a file is being read, if we can't find a proper PK.. signature in the right places we report a format error.

If there is added garbage at the beginning of a Zip file (as inserted by some viruses), you can find out about it, but Archive::Zip will ignore it, and you can still use the archive. When it gets written back out the added stuff will be gone.

There are two ready-to-use utilities in the examples directory that can be used to test file integrity, or that you can use as examples for your own code:

examples/zipcheck.pl shows how to use an attempted extraction to test a file.
examples/ziptest.pl shows how to test CRCs in a file.

Duplicate files in Zip?

Q: Archive::Zip let me put the same file in my Zip twice! Why don't you prevent this?

A: As far as I can tell, this is not disallowed by the Zip spec. If you think it's a bad idea, check for it yourself:

```
  $zip->addFile($someFile, $someName) unless $zip->memberNamed($someName);
```

I can even imagine cases where this might be useful (for instance, multiple versions of files).

File ownership/permissions/ACLS/etc

Q: Why doesn't Archive::Zip deal with file ownership, ACLs, etc.?

A: There is no standard way to represent these in the Zip file format. If you want to send me code to properly handle the various extra fields that have been used to represent these through the years, I'll look at it.

I can't compile but ActiveState only has an old version of Archive::Zip

Q: I've only installed modules using ActiveState's PPM program and repository. But they have a much older version of Archive::Zip than is in CPAN. Will you send me a newer PPM?

A: Probably not, unless I get lots of extra time. But there's no reason you can't install the version from CPAN. Archive::Zip is pure Perl, so all you need is NMAKE, which you can get for free from Microsoft (see the FAQ in the ActiveState documentation for details on how to install CPAN modules).

My JPEGs (or MP3's) don't compress when I put them into Zips!

Q: How come my JPEGs and MP3's don't compress much when I put them into Zips?

A: Because they're already compressed.

Under Windows, things lock up/get damaged

Q: I'm using Windows. When I try to use Archive::Zip, my machine locks up/makes funny sounds/displays a BSOD/corrupts data. How can I fix this?

A: First, try the newest version of Compress::Raw::Zlib. I know of Windows-related problems prior to v1.14 of that library.

Zip contents in a scalar

Q: I want to read a Zip file from (or write one to) a scalar variable instead of a file. How can I do this?

A: Use `IO::String` and the `readFromFileHandle()` and `writeToFileHandle()` methods. See `examples/readScalar.pl` and `examples/writeScalar.pl`.

Reading from streams

Q: How do I read from a stream (like for the Info-Zip `funzip` program)?

A: This is not currently supported, though writing to a stream is.