ADAMS

Advanced **D**ata mining **A**nd **M**achine learning **S**ystem

Module: adams-compress



Peter Reutemann

November 21, 2026

©2012-2025 THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato


Except where otherwise noted, this work is licensed under
http://creativecommons.org/licenses/by-sa/4.0/

Contents

1 Introduction **5**

2 Boolean conditions **7**

3 Compression **9**

4 Decompression **11**

5 Incremental creation **13**

Bibliography **15**

Chapter 1

Introduction

Compressing and decompressing files, archiving multiple files, extracting files from archives. These are all actions that are very common and most of the time manually performed. Using ADAMS, these steps can be automated thanks to the various transformers available that handle archives.

In general, there are two different kinds of actors:

- multi-file archives (compress/decompress)
- single-file algorithms (compression/decompression)

The latter only work on a single file, like for instance gzip. ZIP, on the other hand, works with multiple files. The single-file algorithms can compress/decompress byte arrays in addition to files.

The following chapters cover the actors for compression and decompression in more detail.

Chapter 2

Boolean conditions

The following boolean flow conditions can be applied to file (String/File objects) or byte arrays:

- IsBzip2Compressed
- IsGzipCompressed
- IsRarCompressed
- IsXzCompressed
- IsZipCompressed
- IsZstdCompressed

Chapter 3

Compression

In order to compress one or more files, you can use the following transformers:

- **single files/byte arrays**
  - Bzip2 (see [2])
  - GZIP (see [3])
  - Lzf (see [4])
  - Lzma (see [5])
  - Xz (see [8])
  - Zstd (see [10])
- **multiple files**
  - Tar (see [7])
  - ZIP (see [9])

Single file
These transformers take a single file as input, which gets compressed.[1] You have the choice of selecting a target file. If not, then the generated archive gets placed in the same directory as the input file. Optionally, you can also remove the original file. This is useful if you simply want to save disk-space and compress all files in a directory, but don't need the original files anymore.

Byte array
Byte arrays can be compressed using these transformers as well, generating a compressed byte array.[2]

Multiple files
Transformers that manage archives instead of only compressing single files, you need to supply them with an array of file names that should get added to the archive (no incremental adding possible at the moment). Using the *stripPath* regular expression, you can exert control over what of the (most likely absolute) path of the file names should end up in the archive. In order to strip the complete path from the names, simply use ".*" as expression.[3]

[1] adams-compress-gzip_single_file.flow
[2] adams-compress-byte_array_handling.flow
[3] adams-compress-generate_zip.flow

Chapter 4

Decompression

In order to decompress a compressed file or extract one more files from a multi-file archive, you have the following transformers available:

- **single file archives/compressed byte arrays**
  - UnBzip2 (see [2])
  - UnGZIP (see [3])
  - UnLzf (see [4])
  - UnLzma (see [5])
  - UnXz (see [8])
  - UnZstd (see [10])
- **multi-file archives**
  - UnRAR (see [6])
  - UnTar (see [7])
  - UnZIP (see [9])

Single file archives
When using transformers for decompressing the content of a single file archive, you can choose whether you would like to extract the content to a different directory and even whether to use a different file name (by default, the file name is the one with the compression's suffix).[1]

Compressed byte array
Compressed byte arrays can be decompressed using these transformers as well, restoring the original byte array.[2]

Multi-file archives
By default, all files from a multi-file archive get extracted. If you only require a subset of the files, you can use the *regExp* option to limit the files extracted to the ones that match the regular expression (you can also invert the matching sense).[3] Optionally, you can choose whether to re-create the directory structure stored in the archive.

[1] adams-compress-decompress_gzipped_file.flow
[2] adams-compress-byte_array_handling.flow
[3] adams-compress-extract_from_zip.flow

Chapter 5

Incremental creation

It is also possible to create multi-file archives in stages using the following actors:

- `NewArchive` – initializes the archive and forwards a data structure to be used by the following two actors
- `AppendArchive` – for adding a file or an item from storage to the archive, can be used multiple times
- `CloseArchive` – finalizes and closes the archive

The following archive managers are available:

- `TarArchiveManager` – for tar files (various compressions are supported)
- `ZipArchiveManager` – for zip files

The following examples flows demonstrate the functionality:

- `tar` – *adams-compress-incremental_tar.flow*
- `zip` – *adams-compress-incremental_zip.flow*

Bibliography

[1] *ADAMS* – Advanced Data mining and Machine learning System
`https://adams.cms.waikato.ac.nz/`

[2] *bzip2* – Burrows-Wheeler algorithm
`http://en.wikipedia.org/wiki/Bzip2`

[3] *Gzip* – GNU zip
`http://en.wikipedia.org/wiki/Gzip`

[4] *LZF*
`http://en.wikibooks.org/wiki/Data_Compression/Dictionary_compression#LZF`

[5] *LZMA* – Lempel-Ziv-Markov chain algorithm
`http://en.wikipedia.org/wiki/LempelZivMarkov_chain_algorithm`

[6] *RAR* – proprietary archive file format
`https://en.wikipedia.org/wiki/RAR_(file_format)`

[7] *Tar* – tape archive file format
`http://en.wikipedia.org/wiki/Tar_%28file_format%29`

[8] *XZ Utils* – a set of free command-line lossless data compressors
`https://en.wikipedia.org/wiki/XZ_Utils`

[9] *ZIP* – ZIP file format
`http://en.wikipedia.org/wiki/Zip_%28file_format%29`

[10] *Zstandard* – a lossless data compression algorithm developed by Yann Collet at Facebook
`https://en.wikipedia.org/wiki/Zstd`