

# Zip Plugin 1.1

---

Liest und schreibt Zip-Archive mit Hollywood

**Andreas Falkenhahn**

---



# Inhaltsverzeichnis

<b>1</b>	<b>Allgemeine Informationen</b>	<b>1</b>
1.1	Einführung	1
1.2	Lizenzvereinbarung	1
1.3	Voraussetzung	2
1.4	Installation	2
<b>2</b>	<b>Über zip.hwp</b>	<b>3</b>
2.1	Danksagungen	3
2.2	Häufig gestellte Fragen	3
2.3	Bekannte Probleme	3
2.4	Zukunft	3
2.5	Geschichte	4
<b>3</b>	<b>Benutzung</b>	<b>5</b>
3.1	Aktivieren des Plugins	5
3.2	Zip Archive als Verzeichnisse	5
3.3	Dateien entpacken	6
3.4	Zip-Archiv Grundlagen	7
3.5	Zip-Archive erstellen	7
3.6	Dateien verknüpfen	7
<b>4</b>	<b>Befehlsreferenz</b>	<b>9</b>
4.1	zip.AddDirectory	9
4.2	zip.AddFile	9
4.3	zip.CloseArchive	11
4.4	zip.DeleteFile	12
4.5	zip.ExtractFile	13
4.6	zip.GetFileAtIndex	14
4.7	zip.GetFilesAttributes	14
4.8	zip.GetFileComment	15
4.9	zip.GetObjectType	16
4.10	zip.LocateFile	17
4.11	zip.OpenArchive	17
4.12	zip.RenameFile	18
4.13	zip.SetDefaultPassword	19
4.14	zip.SetFileComment	19
4.15	zip.SetFileCompression	20
4.16	zip.SetFileEncryption	21
4.17	zip.SetFileTime	22

<b>Anhang A Lizenzen</b> .....	<b>23</b>
A.1 LibZip Lizenz .....	23
A.2 AES-Verschlüsselungsunterstützung .....	23
<b>Index</b> .....	<b>25</b>

# 1 Allgemeine Informationen

## 1.1 Einführung

Mit diesem Plugin können Hollywood-Skripte Zip-Archive lesen und schreiben. Es verwendet die neuen Plug-In-Schnittstellen für Datei- und Verzeichnisadapter von Hollywood 6.0, mit denen Sie Zip-Archive durchlaufen können, als wären sie normale Verzeichnisse. Auf Dateien in Zip-Archiven kann auch zugegriffen werden, als wäre das Zip-Archiv ein normales Verzeichnis. Es ist nicht notwendig, eine in einem Zip-Archiv gespeicherte Datei in eine temporäre Datei zu entpacken, bevor sie geöffnet werden kann. Die Adapter-Plugin-Schnittstelle von Hollywood 6.0 ermöglicht das direkte Lesen und Schreiben vom Zip-Archiv in den jeweiligen Datei-Handler.

Darüber hinaus bietet zip.hwp eine Reihe von Befehlen zum Lesen, Ändern und Schreiben von Zip-Archiven. Neue Zip-Archive können erstellt werden, vorhandene Zip-Archive können geöffnet und geändert werden. Es gibt eine Vielzahl von Befehlen, mit denen Sie Attribute von Dateien, die in Zip-Archiven gespeichert sind, lesen, ändern und schreiben können. Darüber hinaus unterstützt zip.hwp das Lesen und Speichern passwortgeschützter Dateien mit starker AES-128-, AES-192- und AES-256-Verschlüsselung.

## 1.2 Lizenzvereinbarung

zip.hwp ist © Copyright 2014-2018 bei Andreas Falkenhahn (im folgenden "der Autor" genannt). Alle Rechte vorbehalten.

Das Programm wird zur Verfügung gestellt "wie es ist" und der Autor kann für keinerlei Schäden, welcher Natur sie auch immer sein mögen, verantwortlich gemacht werden. Sie benutzen dieses Programm völlig auf eigene Gefahr und eigenes Risiko. Der Autor gibt keinerlei Garantien in Verbindung mit der Benutzung dieses Programmes, nicht einmal die Garantie der Funktionstüchtigkeit.

zip.hwp kann frei weitergegeben werden solange die folgenden drei Bedingungen erfüllt sind:

1. Es dürfen keine Änderungen am Programm vorgenommen werden.
2. Das Programm darf nicht verkauft werden.
3. Wenn Sie das Programm auf einer Coverdisk veröffentlichen möchten, müssen Sie erst um Erlaubnis fragen.

Dieses Programm benutzt libzip Copyright (C) 1999-2016 Dieter Baron und Thomas Klausner. Siehe [Abschnitt A.1 \[Libzip-Lizenz\]](#), Seite 23, für Details.

AES-Verschlüsselungsunterstützung basiert auf Code Copyright (C) 2002 Dr Brian Gladman. Siehe [Abschnitt A.2 \[AES-Verschlüsselungsunterstützung\]](#), Seite 23, für Details.

Alle Warenzeichen sind Eigentum ihrer jeweiligen Firmen.

FÜR DIESES PROGRAMM GIBT ES KEINE GARANTIE, SOWEIT ES DIE ANZUWENDENDEN GESETZE ZULASSEN. SOFERN ANDERSWO NICHTS GEGENTEILIGES GESCHRIEBEN STEHT STELLEN DER AUTOR UND/ODER DRITTE DAS PROGRAMM "SO WIE ES IST" ZUR VERFÜGUNG, OHNE IRGEND-EINE GARANTIE, WEDER DIREKT NOCH INDIREKT. DIES BEINHÄLTET, IST ABER NICHT DARAUF BESCHRÄNKT, VERKÄUFLICHKEIT UND EIGNUNG

FÜR EINEN BESTIMMTEN VERWENDUNGSZWECK. DAS VOLLSTÄNDIGE RISIKO DER QUALITÄT UND AUSFÜHRBARKEIT DES PROGRAMMS LIEGT BEIM ANWENDER. SOLLTE SICH DAS PROGRAMM ALS DEFEKT HERAUSSTELLEN, LIEGEN ALLE KOSTEN FÜR SERVICE, INSTANDSETZUNG ODER NACHBESSERUNG BEIM ANWENDER.

KEIN COPYRIGHT-INHABER ODER DRITTER, DER DAS PROGRAMM WIE OBEN ERLAUBT WEITERVERKAUFT, KANN FÜR SCHÄDEN IRGENDWELCHER ART HAFTBAR GEMACHT WERDEN (DIES BEINHÄLTET, IST ABER NICHT BESCHRÄNKT AUF, DATENVERLUST INFOLGE UNFÄHIGKEIT DES PROGRAMMS, MIT ANDEREN PROGRAMMEN ZUSAMMENZUARBEITEN), SELBST WENN EIN SOLCHER INHABER ODER DRITTER AUF DIE MÖGLICHKEIT EINES SOLCHEN SCHADENS HINGEWIESEN WURDE, AUSSER ES BESTEHT EINE SCHRIFTLICHE EINWILLIGUNG ODER WIRD VOM GESETZ VERLANGT.

### 1.3 Voraussetzung

- Minimum: Hollywood 6.0 oder besser
- Unicode: Hollywood 7.0 oder besser wird empfohlen, weil zip.hwp intern UTF-8 für alle Zeichenketten verwendet und Hollywood-Versionen vor 7.0 sind nicht Unicode-fähig

### 1.4 Installation

Die Installation von zip.hwp ist unkompliziert und einfach: Kopieren Sie einfach die Datei `zip.hwp` für die Plattform Ihrer Wahl in das Hollywood Plugin-Verzeichnis. Auf allen Systemen außer auf AmigaOS und kompatiblen Systemen müssen Plugins in einem Verzeichnis mit dem Namen `Plugins` gespeichert werden, das sich im selben Verzeichnis wie das Hauptprogramm von Hollywood befindet. Auf AmigaOS und kompatiblen Systemen müssen Plugins stattdessen in `LIBS: Hollywood` installiert werden. Unter Mac OS X muss sich das Verzeichnis `Plugins` im Verzeichnis `Resources` des Programmpaket befinden, d.h. im Verzeichnis `HollywoodInterpreter.app/Contents/Resources`. Beachten Sie, dass `HollywoodInterpreter.app` im Programmpaket `Hollywood.app` selbst gespeichert ist, nämlich in `Hollywood.app/Contents/Resources`.

Unter Windows sollten Sie auch die Datei `zip.chm` in das Verzeichnis `Docs` Ihrer Hollywood-Installation kopieren. Dann können Sie die Online-Hilfe aufrufen, indem Sie F1 drücken, wenn sich der Cursor in der Hollywood-IDE über einem zip.hwp-Befehl befindet.

Kopieren Sie unter Linux und Mac OS das Verzeichnis `zip`, welches sich im Verzeichnis `Docs` des zip.hwp-Distributionsarchivs befindet, in das Verzeichnis `Docs` Ihrer Hollywood-Installation. Beachten Sie, dass sich das Verzeichnis `Docs` unter Mac OS im Programmpaket `Hollywood.app` befindet, d.h. in `Hollywood.app/Contents/Resources/Docs`.

## 2 Über zip.hwp

### 2.1 Danksagungen

zip.hwp wurde von Andreas Falkenhahn geschrieben. Die Arbeit an diesem Projekt wurde Anfang 2014 als Demonstration der leistungsfähigen neuen Datei- und Verzeichnisadapter-Plugin-Schnittstelle von Hollywood 6.0 gestartet, die es Plugins ermöglichen, sich in Hollywoods Datei- und Verzeichnishandler einzuklinken. zip.hwp nutzt diese Funktion, indem Hollywood denkt, dass Zip-Archive nur Verzeichnisse sind. So ist es möglich, sie mit Hollywoods normalen Verzeichnisfunktionen zu durchlaufen oder Dateien in Zip-Archiven zu öffnen, ohne sie zuerst zu entpacken.

Ein besonderer Dank geht an Helmut Haake und Dominic Widmer für die Übersetzung des Handbuchs ins Deutsche. Fehler oder Verbesserungsvorschläge bzgl. des deutschen Handbuchs bitte an das Übersetzungsteam richten, welches unter [handbuch@gmx.ch](mailto:handbuch@gmx.ch) erreicht werden kann.

Wenn Sie mich kontaktieren möchten, können Sie entweder eine E-Mail an [andreas@airsoftsoftwair.de](mailto:andreas@airsoftsoftwair.de) senden oder das Kontaktformular auf <http://www.hollywood-mal.com> verwenden.

### 2.2 Häufig gestellte Fragen

In diesem Abschnitt werden einige häufig gestellte Fragen behandelt. Bitte lesen Sie sie zuerst, bevor Sie in der Mailingliste oder im Forum nachfragen, da Ihr Problem hier möglicherweise behandelt wurde.

**Q: Wo kann ich um Hilfe bitten?**

A: Es gibt ein lebhaftes Forum auf <http://forums.hollywood-mal.com> und wir haben auch eine Mailing-Liste, die Sie unter [airsoft\\_hollywood@yahogroups.com](mailto:airsoft_hollywood@yahogroups.com) aufrufen können. Besuchen Sie <http://www.hollywood-mal.com> für Informationen darüber, wie Sie der Mailingliste beitreten können.

**Q: Ich habe einen Fehler gefunden.**

A: Bitte informiere mich darüber in den speziellen Bereichen des Forums oder der Mailingliste.

### 2.3 Bekannte Probleme

Hier ist eine Liste von Dingen, die zip.hwp noch nicht unterstützt oder die auf irgendeine Weise verwirrend sein können:

- tbd

### 2.4 Zukunft

Hier sind einige Dinge, die auf meiner Liste stehen:

- tbd

Zögern Sie nicht, mich zu kontaktieren, wenn in zip.hwp eine bestimmte Funktion fehlt, die für Ihr Projekt wichtig ist.

## 2.5 Geschichte

Bitte schauen Sie in die auf englisch verfasste Datei `history.txt`. Hier finden Sie ein vollständiges Änderungsprotokoll von zip.hwp.



## 3 Benutzung

### 3.1 Aktivieren des Plugins

Es gibt zwei Möglichkeiten, dieses Plugin zu verwenden: Sie können das Plugin global aktivieren, indem Sie den Tag `InstallAdapter` auf `True` setzen, wenn es erforderlich ist. Geben Sie dazu einfach die folgende Präprozessor-Anweisung oben im Skript ein:

```
@REQUIRE "zip", {InstallAdapter = True}
```

Wenn Sie das Plugin über `@REQUIRE` aktivieren, wird es global verfügbar und alle folgenden Befehle, die sich mit Dateien befassen, unterstützen das Öffnen von Dateien aus Zip-Archivquellen. Zum Beispiel könnten Sie dann folgende Zeile schreiben:

```
LoadBrush(1, "test.zip/testpicture.jpg")
```

Wenn Sie nur sehr wenige Dateien aus ZIP-Archivquellen öffnen müssen, können Sie das Plugin auch nicht global aktivieren, indem Sie den Tag `InstallAdapter` bei `@REQUIRE` weglassen und einfach den von den meisten Hollywood-Befehlen angebotenen entsprechenden Tag `Adapter` verwenden, um die Datei mit dem `zip.hwp`-Plugin zu öffnen. Hier ist ein Beispiel:

```
LoadBrush(1, "test.zip/testpicture.jpg", {Adapter = "zip"})
```

Mit dem Tag `Adapter` wird `LoadBrush()` angewiesen, die angegebene Datei mit dem angegebenen Adapter zu öffnen, der in unserem Fall `"zip"` ist. Mit dem Tag `Adapter` können Sie dieses Plugin also auch verwenden, ohne zuvor einen globalen Dateiadapter installiert zu haben.

Dasselbe gilt für Hollywood-Befehle, die mit Verzeichnissen arbeiten. Sobald das Zip-Plugin aktiviert wurde, indem `InstallAdapter` auf `True` gesetzt wird, ist es möglich, folgendes zu verwenden:

```
OpenDirectory(1, "test.zip")
```

Sie könnten dann alle Dateien und Verzeichnisse in `test.zip` durchlaufen. Wenn Sie den globalen Adapter für `zip.hwp` nicht aktiviert haben, verwenden Sie einfach den lokalen `Adapter` wie oben, z.B:

```
OpenDirectory(1, "test.zip", {Adapter = "zip"})
```

Weitere Informationen über die Arbeit mit Zip-Archiven als Verzeichnisse finden Sie im nächsten Kapitel.

### 3.2 Zip Archive als Verzeichnisse

Wenn Sie den Tag `InstallAdapter` auf `True` setzen, hängt sich das Zip-Plugin in Hollywoods Verzeichnis-Handler ein, damit Hollywood glaubt, dass Zip-Archive normale Verzeichnisse sind. Auf diese Weise können Sie alle Dateien und Verzeichnisse in einem Zip-Archiv mithilfe normaler Befehle aus der DOS-Bibliothek von Hollywood durchlaufen.

Um beispielsweise alle Dateien und Verzeichnisse in einer Datei mit dem Namen `test.zip` zu durchlaufen, können Sie den folgenden Code verwenden:

```
OpenDirectory(1, "test.zip")
Local e = NextDirectoryEntry(1)
While e <> Nil
```

```

    DebugPrint(e.name)
    e = NextDirectoryEntry(1)
Wend
CloseDirectory(1)

```

Wenn Sie nicht vom Stammverzeichnis innerhalb von `test.zip` aus starten möchten, können Sie auch bequem von einem Unterverzeichnis aus starten, indem Sie einfach vorgeben, dass `test.zip` ein Verzeichnis ist. Z.B. um auf ein Unterverzeichnis mit dem Namen `files` in `test.zip` zuzugreifen, gehen Sie wie folgt vor:

```
OpenDirectory(1, "test.zip/files")
```

Schließlich ist es auch möglich, rekursiv alle Dateien und Verzeichnisse in einem Zip-Archiv zu durchlaufen. Hier ist eine Funktion, die das tut:

```

Function p_DumpZip(d$, idt)
    Local id = OpenDirectory(1, d$)
    Local e = NextDirectoryEntry(id)
    While e <> Nil
        If e.Type = #DOSTYPE_DIRECTORY
            DebugPrint(RepeatStr(" ", idt) .. "+", e.name)
            p_DumpZip(FullPath(d$, e.name), idt + 4)
        Else
            DebugPrint(RepeatStr(" ", idt) .. ",e.name,e.size,e.time)
        EndIf
        e = NextDirectoryEntry(id)
    Wend
    CloseDirectory(id)
EndFunction

```

Um den Inhalt eines Zip-Archivs abzulegen, rufen Sie einfach folgende Funktion auf:

```
p_DumpZip("test.zip", 0)
```

Es wird dann einen schönen Baum des Inhalts des Zip-Archivs ausgegeben.

### 3.3 Dateien entpacken

Da sich `zip.hwp` in den Datei-Handler von Hollywood einklinkt, ist das Entpacken von Dateien nur eine Frage der Verwendung des Hollywood-Befehls `CopyFile()` für die Datei, die Sie entpacken möchten. Um beispielsweise eine Datei mit dem Namen `testpicture.jpg` aus `test.zip` zu entpacken, verwenden Sie einfach die folgende Zeile:

```
CopyFile("test.zip/testpicture.jpg", "outputdir")
```

Da `CopyFile()` auch ganze Verzeichnisse einschließlich aller Unterverzeichnisse kopieren kann und weil sich `zip.hwp` auch in den Datei-Handler von Hollywood einhakt, ist es sogar möglich, ein ganzes Archiv mit `CopyFile()` so zu entpacken:

```
CopyFile("test.zip", "outputdir")
```

Dies wird jedoch ziemlich langsam sein, da `zip.hwp` das Zip-Archiv für jede einzelne Datei öffnet und schließt, die enpackt werden muss, was natürlich ein Performance-Killer ist. Deshalb bietet `zip.hwp` auch einen dedizierten Befehl zum Entpacken von Dateien für die Feinsteuerung von Zip-Archiven. Siehe [Abschnitt 4.5 \[zip.ExtractFile\]](#), Seite 13, für Details.

### 3.4 Zip-Archiv Grundlagen

Zip-Archive sind nur eine Sammlung von Dateien, die in Indizes von 0 bis zur Anzahl der Einträge im Zip-Archiv minus 1 gespeichert werden. Es ist nicht notwendig, Verzeichnisse als einzelne Einträge zu speichern. Stattdessen können sie auch als Teil eines Dateinamens gespeichert werden. Z.B. wenn eine Datei im Zip-Archiv als `a/b/c/test.txt` gespeichert ist, werden die Verzeichnisse `a`, `b` und `c` implizit als existierend deklariert, obwohl sie keine eigenen Einträge in der Datei im Zip-Archiv haben, sondern nur als Teil einer Datei existieren.

Natürlich können Verzeichnisse auch als einzelne Einträge anstatt als Teil eines Dateinamens gespeichert werden. In diesem Fall werden sie einfach als Dateien mit einer Größe von 0 Bytes gespeichert, wobei der Dateiname mit einem Schrägstrich endet, der anzeigt, dass der Eintrag ein Verzeichnis ist. Da es in Zip-Archiven keinen eindeutigen Verzeichniseintragstyp gibt, können alle Befehle in diesem Plugin, die sich mit Dateien befassen, auch auf Verzeichnisse innerhalb des Zip-Archivs zugreifen. Beachten Sie also, dass ein Befehl wie `zip.RenameFile()` auch zum Umbenennen von Verzeichnissen verwendet werden und `zip.DeleteFile()` auch Verzeichnisse löschen kann. Innerhalb eines Zip-Archivs sind Verzeichnisse und Dateien ziemlich identisch, nur dass bei Verzeichnissen ihr Dateiname in einem Schrägstrich endet, um anzuzeigen, dass es sich nicht um eine Datei handelt.

### 3.5 Zip-Archive erstellen

Mit den Befehlen `zip.OpenArchive()`, `zip.AddFile()` und `zip.CloseArchive()` können Sie neue Zip-Archive erstellen. Der folgende Code zeigt, wie ein neues Zip-Archiv mit dem Namen `test.zip` erstellt wird, das die Datei `testpicture.jpg` enthält:

```
zip.OpenArchive(1, "test.zip", #MODE_WRITE)
zip.AddFile(1, "testpicture.jpg")
zip.CloseArchive(1)
```

Beachten Sie, dass `zip.AddFile()` die Datei nicht sofort komprimiert und in das Archiv schreibt. Stattdessen werden Dateien zuerst gesammelt und erst komprimiert und in das Archiv geschrieben, wenn Sie `zip.CloseArchive()` aufrufen. Aus diesem Grund kann das Schließen eines Archivs einige Zeit in Anspruch nehmen. Darum gibt es auch die Möglichkeit, eine Callback-Funktion zu übergeben, die von `zip.CloseArchive()` von Zeit zu Zeit aufgerufen wird, so dass Sie eine Statusleiste oder etwas anderes aktualisieren können.

### 3.6 Dateien verknüpfen

Beachten Sie, dass alle in den Präprozessor-Anweisungen deklarierten Dateien automatisch mit Ihrem Applet oder Ihrem Programm verknüpft werden, wenn sich Hollywood im Kompilierungsmodus befindet. Wenn Sie also etwas wie die folgende Zeile verwenden, wird nicht nur `testpicture.jpg` sondern das gesamte Zip-Archiv `test.zip` mit Ihrem Applet oder Programm verknüpft:

```
@BRUSH 1, "test.zip/testpicture.jpg"
```

Wenn Sie das nicht möchten, können Sie den optionalen Tag `Link` auf `False` setzen. Wenn `Link` auf `False` gesetzt ist, wird Hollywood die angegebene Datei nicht mit Ihrem Applet oder Programm verknüpfen. Dies bedeutet jedoch, dass Sie `test.zip` mit Ihrem Applet

oder Programm verteilen müssen, damit die Daten daraus geladen werden können. So deaktivieren Sie die Verknüpfung:

```
@BRUSH 1, "test.zip/testpicture.jpg", {Link = False}
```

Wenn Sie so vorgehen, wird Hollywood die Datei niemals mit Ihrem Applet oder Programm verknüpfen. Stattdessen wird es immer aus der angegebenen Datei geladen.

## 4 Befehlsreferenz

### 4.1 zip.AddDirectory

#### BEZEICHNUNG

zip.AddDirectory – fügt ein Verzeichnis zum Zip-Archiv hinzu

#### ÜBERSICHT

```
idx = zip.AddDirectory(id, d$)
```

#### BESCHREIBUNG

Dieser Befehl erstellt ein neues Verzeichnis mit dem Namen `d$` im Zip-Archiv mit der ID `id` und gibt den Index in `idx` zurück. Das Verzeichnis ist leer und Sie können Dateien mit dem Befehl `zip.AddFile()` hinzufügen.

#### EINGABEN

`id`            Identifikator des zu verwendenden Zip-Archivs  
`d$`            Name des Verzeichnisses, das im ZIP-Archiv erstellt werden soll

#### RÜCKGABEWERTE

`idx`            Index des neu hinzugefügten Verzeichnisses im ZIP-Archiv

### 4.2 zip.AddFile

#### BEZEICHNUNG

zip.AddFile – fügt eine Datei zum Zip-Archiv hinzu

#### ÜBERSICHT

```
idx = zip.AddFile(id, f$[, table])
```

#### BESCHREIBUNG

Dieser Befehl fügt die durch `f$` angegebene Datei dem in `id` angegebenen Zip-Archiv hinzu und gibt den Index der neu hinzugefügten Datei in `idx` zurück. Mit dem optionalen Tabellenargument `table` können Sie weitere Optionen angeben.

Die folgenden Tags werden derzeit vom optionalen Argument `table` erkannt:

**newName:** Mit diesem Tag können Sie diese Datei unter einem neuen Namen im Zip-Archiv speichern. Wenn Sie die Datei in einem Unterverzeichnis im ZIP-Archiv speichern möchten, müssen Sie auch diesen Tag verwenden und den Namen des/der Unterverzeichnisse(s) in  **newName** angeben. Wenn  **newName** weggelassen wird, wird die Datei immer im Stammverzeichnis des Zip-Archivs gespeichert.

#### Encryption:

Mit diesem Tag können Sie die gewünschte Verschlüsselungsmethode für die Datei festlegen. Es kann auf eine der folgenden speziellen Konstanten eingestellt werden:

**#ZIP\_EM\_NONE:**

Keine Verschlüsselung. Dies ist der Standardwert.

**#ZIP\_EM\_AES\_128:**  
Winzip AES-128-Verschlüsselung.

**#ZIP\_EM\_AES\_192:**  
Winzip AES-192-Verschlüsselung.

**#ZIP\_EM\_AES\_256:**  
Winzip AES-256-Verschlüsselung.

Wenn Sie den Tag **Encryption** angeben, müssen Sie auch ein Passwort im Tag **Password** angeben (siehe unten), das zum Entschlüsseln der Datei benötigt wird. Wenn Sie den Tag **Password** nicht verwenden, wird das mit `zip.SetDefaultPassword()` festgelegte Standard-Passwort verwendet.

**Password:**

Wenn der Tag **Encryption** auf einen anderen Wert als **#ZIP\_EM\_NONE** gesetzt wurde (siehe oben), kann dieser Tag auf ein Passwort gesetzt werden, welches zum Schutz der Datei verwendet werden soll. Wenn Sie diesen Tag nicht angeben, wird das mit `zip.SetDefaultPassword()` festgelegte Standard-Passwort verwendet.

**Compression:**

Mit diesem Tag kann die gewünschte Komprimierungsmethode für die Datei festgelegt werden. Die folgenden Komprimierungsmethoden werden derzeit unterstützt:

**#ZIP\_CM\_DEFAULT:**  
Dies ist die Standardeinstellung. Derzeit identisch mit **#ZIP\_CM\_DEFLATE**.

**#ZIP\_CM\_STORE:**  
Die Datei unkomprimiert speichern.

**#ZIP\_CM\_BZIP2:**  
Die Datei mit dem bzip2-Algorithmus komprimieren.

**#ZIP\_CM\_DEFLATE:**  
Die Datei mit dem zlib-Algorithmus und den Standardoptionen komprimieren.

Beachten Sie, dass nur **#ZIP\_CM\_DEFLATE** und **#ZIP\_CM\_STORE** als universell unterstützt gelten.

Wenn Sie diesen Tag angeben, können Sie auch den Tag **CompressionFlags** übergeben, um die Komprimierungsstufe festzulegen (siehe unten).

**CompressionFlags:**

Mit diesem Tag kann die Komprimierungsstufe definiert werden. Sie geht von 1 für die schnellste Komprimierung bis 9 für die höchste Komprimierung. Sie können auch 0 übergeben, um die Standardeinstellungen des Komprimierers zu verwenden. Der Standardwert ist 0.

**Comment:** Mit diesem Tag kann die Datei dem Zip-Archiv mit einem angehängten Kommentar hinzugefügt werden.

**Time:** Mit diesem Tag kann der Datumsstempel der Datei geändert werden. Standardmäßig wird der Datumsstempel aus der in `f$` angegebenen Datei genommen. Wenn Sie der Datei einen anderen Datumsstempel zuweisen möchten, müssen Sie diesem Tag eine Zeichenkette im Standard-Hollywood-Datumsformat von `dd-mmm-yyyy hh:mm:ss` übergeben.

Beachten Sie, dass dieser Befehl nicht sofort beginnt, die Datei zu komprimieren und in das Zip-Archiv zu schreiben. Stattdessen wird die Datei nur in eine interne Liste eingefügt und das Komprimieren und Schreiben erfolgt, sobald Sie `zip.CloseArchive()` aufrufen. Das bedeutet, dass Sie sicherstellen müssen, dass die in `f$` angegebene Datei noch verfügbar ist, wenn Sie `zip.CloseArchive()` aufrufen. Beachten Sie deshalb, dass wenn Sie den Namen einer temporären Datei an `f$` übergeben, Sie diese temporäre Datei nicht löschen, bevor Sie `zip.CloseArchive()` aufrufen.

#### EINGABEN

`id` Identifikator des zu verwendenden Zip-Archivs  
`f$` Pfad zu einer Datei, die dem ZIP-Archiv hinzugefügt werden soll  
`table` Optional: Tabelle mit weiteren Optionen (siehe oben)

#### RÜCKGABEWERTE

`idx` Index der neu hinzugefügten Datei im ZIP-Archiv

### 4.3 zip.CloseArchive

#### BEZEICHNUNG

`zip.CloseArchive` – schliesst das Zip-Archiv

#### ÜBERSICHT

`zip.CloseArchive(id[, discard, callback, userdata])`

#### BESCHREIBUNG

Dieser Befehl schließt das in `id` angegebene Zip-Archiv. Beachten Sie, dass `zip.CloseArchive()` den Punkt angibt, an dem das Komprimieren und Schreiben der Daten tatsächlich erfolgt, wenn das Zip-Archiv zum Schreiben geöffnet wurde. Aus diesem Grund kann es einige Zeit dauern, bis dieser Befehl seine Arbeit beendet hat.

Wenn Sie alle Änderungen verwerfen möchten, die am ZIP-Archiv vorgenommen wurden, müssen Sie `True` im Parameter `discard` übergeben. In diesem Fall wird das ursprüngliche Zip-Archiv nicht geändert und alle Änderungen werden verworfen. Dies wird auch mit allen Zip-Archiven geschehen, die Sie mit `zip.OpenArchive()` öffnen, aber vergessen mit `zip.CloseArchive()` zu schließen. Änderungen werden nur dann in das Zip-Archiv geschrieben, wenn Sie explizit `zip.CloseArchive()` aufrufen, wobei `discard False` ist.

Wenn Sie den Fortschritt beim Komprimieren von Daten und beim Schreiben in das Zip-Archiv überwachen möchten, können Sie eine Callback-Funktion im dritten Parameter `callback` übergeben. Optional ist es auch möglich, Benutzerdaten anzugeben, die in ihrem vierten Argument `userdata` an die Callback-Funktion übergeben werden.

Der Parameter `userdata` kann Werte beliebigen Typs annehmen: Zahlen, Zeichenketten, Tabellen und sogar Funktionen können als Benutzerdaten übergeben werden.

Die Status-Callback-Funktion empfängt ein einzelnes Tabellenelement, das die folgenden Felder enthält:

**Action:** Enthält die Zeichenkette "CloseArchive".

**ID:** Enthält den Identifikator des Zip-Archivs, an dem gerade gearbeitet wird.

**Progress:**  
Enthält einen Wert zwischen 0 und 100, der angibt, wie viel Arbeit bereits erledigt wurde.

**UserData:**  
Enthält den Wert, den Sie im Argument `userdata` übergeben haben.

Offensichtlich wird die Callback-Funktion niemals aufgerufen, wenn `discard` auf `True` gesetzt ist.

## EINGABEN

`id` Identifikator des zu schließenden Zip-Archivs

`discard` Optional: `True` um alle Änderungen zu verwerfen, `False`, um alle Änderungen in das Zip-Archiv zu schreiben (voreingestellt ist `False`)

`callback` Optional: Funktion, die von Zeit zu Zeit aufgerufen wird

`userdata` Optional: Benutzerspezifische Daten, die an die Callback-Funktion übergeben werden

## 4.4 zip.DeleteFile

### BEZEICHNUNG

`zip.DeleteFile` – löscht eine Datei aus dem Zip-Archiv

### ÜBERSICHT

`zip.DeleteFile(id, idx)`

### BESCHREIBUNG

Dieser Befehl löscht die Datei im Index `idx` in dem von `id` angegebenen Zip-Archiv.

Beachten Sie, dass die Änderung am Zip-Archiv nicht sofort erfolgt, sondern bis zum Aufruf von `zip.CloseArchive()` verschoben wird.

Dieser Befehl kann auch für Verzeichnisse verwendet werden. Siehe [Abschnitt 3.4 \[Zip Archiv-Grundlagen\]](#), Seite 7, für Details.

### EINGABEN

`id` Identifikator des zu verwendenden Zip-Archivs

`idx` Zu löschende Datei



## 4.5 zip.ExtractFile

### BEZEICHNUNG

zip.ExtractFile – entpackt eine Datei aus dem Zip-Archiv

### ÜBERSICHT

```
zip.ExtractFile(id, idx, dst$[, table])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Datei im Index `idx` innerhalb des von `id` angegebenen Zip-Archivs zu der von `dst$` angegebenen externen Datei zu entpacken. Wenn `dst$` bereits existiert, wird sie überschrieben. Mit dem optionalen Tabellenargument `table` können Sie folgende Optionen für die Operation angeben:

#### Password:

Wenn die zu entpackende Datei durch ein Passwort geschützt ist, müssen Sie dieses Passwort hier angeben. Wenn Sie diesen Tag nicht angeben, wird das Standard-Passwort verwendet, das mit dem `zip.SetDefaultPassword()` festgelegt wurde.

#### Callback:

Mit diesem Tag können Sie eine Funktion übergeben, die von Zeit zu Zeit aufgerufen werden soll. Dies kann nützlich sein, wenn Sie eine Statusleiste oder etwas anzeigen möchten, während die Zip-Datei entpackt wird. Die Funktion erhält eine Tabelle als einziges Argument. In der Tabelle werden folgende Felder initialisiert:

**Action:** Enthält die Zeichenkette "ExtractFile".

**ID:** Enthält den Identifikator des Zip-Archivs.

#### Progress:

Enthält einen Wert zwischen 0 und 100, der angibt, wie viel Arbeit bereits erledigt wurde.

#### UserData:

Enthält den Wert, den Sie im Argument `UserData` übergeben haben (siehe unten).

Sie können auch Benutzerdaten, die an Ihren Callback-Funktion weitergeleitet werden, mit dem folgenden Tag übergeben.

#### UserData:

Dieser Tag kann auf beliebige Daten gesetzt werden, die an die Callback-Funktion übergeben werden sollen, den Sie im Tag `Callback` übergeben haben. Wenn Sie diesen Tag ohne den Tag `Callback` angeben, wird er einfach ignoriert.

### EINGABEN

<code>id</code>	Identifikator des zu verwendenden Zip-Archivs
<code>idx</code>	Index der zu entpackenden Datei
<code>dst\$</code>	Gewünschte Zieldatei

`table` Optional: Tabelle mit weiteren Parametern

## 4.6 zip.GetFilesAtIndex

### BEZEICHNUNG

`zip.GetFilesAtIndex` – ermittelt den Namen der Datei im Index

### ÜBERSICHT

```
name$ = zip.GetFilesAtIndex(id, idx)
```

### BESCHREIBUNG

Dieser Befehl gibt den Namen der Datei im Index `idx` in dem von `id` angegebenen Zip-Archiv zurück. Wenn der von diesem Befehl zurückgegebene Name mit einem Schrägstrich endet, handelt es sich um ein Verzeichnis, andernfalls um eine Datei. Siehe [Abschnitt 3.4 \[Zip-Archiv Grundlagen\], Seite 7](#), für Details.

Um die Anzahl der Dateien in einem Zip-Archiv herauszufinden, können Sie `#ZIPATTRNUMENTRIES` mit dem Befehl `GetAttribute()` von Hollywood abfragen. Siehe [Abschnitt 4.9 \[zip.GetObjectType\], Seite 16](#), für Details.

### EINGABEN

`id` Identifikator des zu verwendenden Zip-Archivs  
`idx` Index der Rückfrage (im Bereich von 0 bis Anzahl der Einträge minus 1)

### RÜCKGABEWERTE

`name$` Name des Eintrags im Index

## 4.7 zip.GetFilesAttributes

### BEZEICHNUNG

`zip.GetFilesAttributes` – ermittelt die Dateiattribute

### ÜBERSICHT

```
t = zip.GetFilesAttributes(id, idx)
```

### BESCHREIBUNG

Dieser Befehl gibt Attribute der Datei an dem in `idx` angegebenen Index innerhalb des von `id` angegebenen Zip-Archivs zurück. `zip.GetFilesAttributes()` gibt eine Tabelle mit folgenden Informationen über die Datei zurück:

**Size:** Die Größe der Datei in Byte oder 0 für Verzeichnisse.

**CompressedSize:**  
Die komprimierte Größe der Datei in Byte oder 0 für Verzeichnisse.

**CRC32:** Die CRC32-Prüfsumme der Datei oder 0 für Verzeichnisse.

**Compression:**  
Die für die Datei verwendete Komprimierungsmethode. Dies wird eine der folgenden speziellen Konstanten sein:

**#ZIP\_CM\_DEFAULT:**  
Standardkomprimierung, derzeit identisch mit **#ZIP\_CM\_DEFLATE**.

**#ZIP\_CM\_STORE:**  
Die Datei ist unkomprimiert.

**#ZIP\_CM\_BZIP2:**  
Die Datei ist mit dem bzip2-Algorithmus komprimiert.

**#ZIP\_CM\_DEFLATE:**  
Die Datei ist mit dem ZLIB-Algorithmus und den Standardoptionen kompromiert.

**Encryption:**

Die für die Datei verwendete Verschlüsselungsmethode. Dies wird eine der folgenden speziellen Konstanten sein:

**#ZIP\_EM\_NONE:**  
Keine Verschlüsselung.

**#ZIP\_EM\_AES\_128:**  
Winzip AES-128 Verschlüsselung.

**#ZIP\_EM\_AES\_192:**  
Winzip AES-192 Verschlüsselung.

**#ZIP\_EM\_AES\_256:**  
Winzip AES-256 Verschlüsselung.

**Time:** Der Datumsstempel für die Datei. Dies wird im Standard-Hollywood-Datumsformat von `dd-mmm-yyyy hh:mm:ss` sein.

**EINGABEN**

**id** Identifikator des zu verwendenden Zip-Archivs

**idx** Index der abzufragenden Datei

**RÜCKGABEWERTE**

**t** Tabelle mit Dateiattributen

## 4.8 zip.GetFilesComment

**BEZEICHNUNG**

`zip.GetFilesComment` – ermittelt den Dateikommentar

**ÜBERSICHT**

`c$ = zip.GetFilesComment(id[, idx])`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um den Kommentar einer Datei innerhalb des von `id` angegebenen Zip-Archivs oder des gesamten Zip-Archivs zu ermitteln. Wenn der optionale Parameter `idx` angegeben wird, ruft `zip.GetFilesComment()` den Kommentar

der Datei aus diesem Index ab. Wenn der Parameter `idx` nicht angegeben oder auf `-1` gesetzt ist, wird der Kommentar des Zip-Archivs selbst zurückgegeben.

Dieser Befehl kann auch auf Verzeichnisse angewendet werden. Siehe [Abschnitt 3.4 \[Zip-Archiv Grundlagen\]](#), Seite 7, für Details.

#### EINGABEN

`id` Identifikator des zu verwendenden Zip-Archivs

`idx` Optional: Index der Datei, deren Kommentar ermittelt werden soll; Wenn dies weggelassen oder auf `-1` gesetzt wird, wird der Kommentar des gesamten Archivs zurückgegeben (voreingestellt ist `-1`).

#### RÜCKGABEWERTE

`c$` Dateikommentar oder leere Zeichenkette, wenn kein Kommentar vorhanden ist

## 4.9 zip.GetObjectType

#### BEZEICHNUNG

`zip.GetObjectType` – ruft den Zip-Archiv-Objektyp ab

#### ÜBERSICHT

```
type = zip.GetObjectType()
```

#### BESCHREIBUNG

Dieser Befehl gibt den Objekttyp zurück, der von Zip-Archiven verwendet wird, die mit dem Befehl `zip.OpenArchive()` geöffnet wurden. Sie können diesen Objekttyp dann mit Befehlen aus der Objektbibliothek von Hollywood wie `GetAttribute()`, `SetObjectData()`, `GetObjectData()` usw. verwenden.

Insbesondere kann der Befehl `GetAttribute()` von Hollywood verwendet werden, um bestimmte Eigenschaften von mit `zip.OpenArchive()` geöffneten Zip-Archiven abzufragen. Die folgenden Attribute werden derzeit von `GetAttribute()` für Zip-Archive unterstützt:

`#ZIPATTRNUMENTRIES`:

Gibt die Anzahl der Einträge im ZIP-Archiv zurück.

#### EINGABEN

Keine

#### RÜCKGABEWERTE

`type` Interner Zip-Archivtyp zur Verwendung mit der Objektbibliothek von Hollywood

#### BEISPIEL

```
zip.OpenArchive(1, "test.zip")
ZIP_ARCHIVE = zip.GetObjectType()
numentries = GetAttribute(ZIP_ARCHIVE, 1, #ZIPATTRNUMENTRIES)
```

Der obige Code öffnet `test.zip` und fragt die Anzahl der Einträge im Archiv über `GetAttribute()` ab.

## 4.10 zip.LocateFile

### BEZEICHNUNG

zip.LocateFile – sucht die Datei im ZIP-Archiv

### ÜBERSICHT

```
idx = zip.LocateFile(id, name$[, table])
```

### BESCHREIBUNG

Dieser Befehl sucht nach der in `name$` angegebene Datei innerhalb des von `id` angegebenen Zip-Archivs und gibt ihren Index zurück, wenn sie gefunden wird, andernfalls wird -1 zurückgegeben.

Mit dem optionalen Argument `table` können weitere Optionen angegeben werden. Folgende Tabellentags werden derzeit erkannt:

- NoCase:** Wenn dieser Tag auf `True` festgelegt ist, unterscheidet `zip.LocateFile()` nicht zwischen Groß- und Kleinbuchstaben. Dies macht die Suche langsamer. Der Standardwert ist `False`.
- NoDir:** Wenn dieser Tag auf `True` gesetzt ist, stimmt `zip.LocateFile()` nur mit dem Dateinamen überein. Ausserdem wird es auch auf `True` gesetzt, wenn sich die Datei in einem Unterverzeichnis des Archivs befindet. Der Standardwert ist `False`.

Dieser Befehl kann auch auf Verzeichnisse angewendet werden. Siehe [Abschnitt 3.4 \[Zip-Archiv Grundlagen\]](#), Seite 7, für Details.

### EINGABEN

- `id` Identifikator des zu verwendenden Zip-Archivs
- `name$` Name der zu suchenden Datei
- `table` Optional: Tabellenargument mit weiteren Optionen (siehe oben)

### RÜCKGABEWERTE

- `idx` Index der Datei innerhalb des Zip-Archivs oder -1, wenn es nicht gefunden wurde

## 4.11 zip.OpenArchive

### BEZEICHNUNG

zip.OpenArchive – öffnet ein Zip-Archiv zum Lesen oder Schreiben

### ÜBERSICHT

```
[id] = zip.OpenArchive(id, filename$[, mode])
```

### BESCHREIBUNG

Dieser Befehl versucht, das durch `filename$` angegebene Zip-Archiv zu öffnen und weist ihr den Identifikator `id` zu. Wenn Sie `Nil` in `id` übergeben, wählt `zip.OpenArchive()` automatisch eine freie ID aus und gibt sie zurück. Wenn die Datei nicht existiert, schlägt dieser Befehl fehl, es sei denn, Sie verwenden das Argument `mode`, um ein Zip-Archiv zum Schreiben zu öffnen. In diesem Fall erstellt `zip.OpenArchive()` die Datei für Sie.

Die folgenden Modi werden derzeit unterstützt:

**#MODE\_READ:**

Öffnet das Zip-Archiv zum Lesen. Dies ist der Standardmodus.

**#MODE\_READWRITE:**

Öffnet das Zip-Archiv zum Lesen und Schreiben. Wenn das angegebene Zip-Archiv nicht existiert, wird es automatisch erstellt.

**#MODE\_WRITE:**

Öffnet das Zip-Archiv zum Schreiben. Wenn das angegebene Zip-Archiv bereits existiert, wird es überschrieben.

Obwohl `zip.hwp` beim Beenden des Programms alle geöffneten Zip-Archive automatisch schließt, wird dringend empfohlen, ein geöffnetes Zip-Archiv nach Abschluss der Arbeit mit dem Befehl `zip.CloseArchive()` zu schliessen. Andernfalls verschwenden Sie Ressourcen. Falls Sie ein Zip-Archiv schreiben oder ändern, ist `zip.CloseArchive()` der Moment, bei dem die eigentliche Arbeit erledigt wird.

Beachten Sie, dass `zip.OpenArchive()` ein Standard-Hollywood-Objekt erstellt, das auch mit Befehlen aus Hollywoods Objektbibliothek wie `GetAttribute()`, `SetObjectData()`, `GetObjectData()` usw. verwendet werden kann. Siehe [Abschnitt 4.9 \[zip.GetObjectType\]](#), [Seite 16](#), für Details.

#### EINGABEN

**id** Identifikator der Datei oder `Nil` für die automatische ID-Auswahl

**filename\$** Name der zu öffnenden Datei

**mode** Modus, um die Datei zu öffnen; kann `#MODE_READ`, `#MODE_WRITE` oder `#MODE_READWRITE` sein (der Standardwert ist `#MODE_READ`)

#### RÜCKGABEWERTE

**id** Optional: Identifikator der Datei; wird nur zurückgegeben, wenn Sie `Nil` als Argument in `id` übergeben (siehe oben)

## 4.12 zip.RenameFile

#### BEZEICHNUNG

`zip.RenameFile` – benennt die Datei im Zip-Archiv um

#### ÜBERSICHT

`zip.RenameFile(id, idx, newname$)`

#### BESCHREIBUNG

Dieser Befehl benennt die Datei beim Index `idx` im von `id` angegebenen Zip-Archiv in den Namen um, der im Argument `newname$` übergeben wurde.

Wenn Sie einen Verzeichniseintrag umbenennen müssen, muss `newname$` mit einem Schrägstrich enden. Siehe [Abschnitt 3.4 \[Zip-Archiv Grundlagen\]](#), [Seite 7](#), für Details.

Beachten Sie, dass die Änderung am Zip-Archiv nicht sofort erfolgt, sondern verschoben wird, bis Sie `zip.CloseArchive()` aufrufen.

**EINGABEN**

<code>id</code>	Identifikator des zu verwendenden Zip-Archivs
<code>idx</code>	Datei zum Umbenennen
<code>newname\$</code>	Neuer Name für die Datei

### 4.13 `zip.SetDefaultPassword`

**BEZEICHNUNG**

`zip.SetDefaultPassword` – legt das Standard-Passwort fest

**ÜBERSICHT**

```
zip.SetDefaultPassword(id, pwd$)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um das Standard-Passwort zum Verschlüsseln und Entschlüsseln von Dateien festzulegen und wird benutzt, wenn kein anderes Passwort angegeben wird. Sie müssen den Identifikator eines Zip-Archivs in `id` und ein Passwort in `pwd$` übergeben. Wenn Sie eine leere Zeichenkette in `pwd$` übergeben, wird das Standard-Passwort nicht festgelegt.

Befehle, die das Standard-Passwort verwenden, wenn kein anderes Passwort explizit angegeben wird, sind `zip.AddFile()`, `zip.ExtractFile()`, und `zip.SetFileEncryption()`.

**EINGABEN**

<code>id</code>	Identifikator des zu verwendenden Zip-Archivs
<code>pwd\$</code>	Neues Standard-Passwort oder leere Zeichenkette, um das Standard-Passwort aufzuheben

### 4.14 `zip.SetFileComment`

**BEZEICHNUNG**

`zip.SetFileComment` – legt den Dateikommentar fest

**ÜBERSICHT**

```
zip.SetFileComment(id[, c$, idx])
```

**BESCHREIBUNG**

Mit diesem Befehl kann der Kommentar für eine Datei innerhalb des von `id` angegebenen Zip-Archivs oder für das gesamte Zip-Archiv gesetzt werden. Wenn der optionale Parameter `idx` angegeben wird, wird der durch `c$` angegebene Kommentar für die Datei dieses Indexes festgelegt. Wird der optionale Parameter `idx` weggelassen oder auf `-1` gesetzt, wird der durch `c$` angegebene Kommentar für das gesamte Zip-Archiv gesetzt. Sie können den Kommentar einer Datei oder des gesamten Archivs auch entfernen, indem Sie das Argument `c$` weglassen oder auf eine leere Zeichenkette setzen.

Beachten Sie, dass die Änderung am Zip-Archiv nicht sofort erfolgt, sondern verschoben wird, bis Sie `zip.CloseArchive()` aufrufen.

Dieser Befehl kann auch auf Verzeichnisse angewendet werden. Siehe [Abschnitt 3.4 \[Zip-Archive Grundlagen\], Seite 7](#), für Details.

#### EINGABEN

<code>id</code>	Identifikator des zu verwendenden Zip-Archivs
<code>c\$</code>	Optional: legt den Kommentar fest oder der Kommentar wird mit einer leeren Zeichenkette entfernt
<code>idx</code>	Optional: Index der Datei, deren Kommentar gesetzt werden soll; Wenn dies weggelassen oder auf -1 gesetzt wird, wird der Kommentar für das gesamte Archiv gesetzt (standardmäßig -1)

## 4.15 zip.SetFileCompression

#### BEZEICHNUNG

`zip.SetFileCompression` – legt die Dateikomprimierung fest

#### ÜBERSICHT

```
zip.SetFileCompression(id, idx, method[, flags])
```

#### BESCHREIBUNG

Dieser Befehl legt die Komprimierungsmethode für die Datei im Index `idx` in dem in `id` angegebenen Zip-Archiv auf die in `method` angegebene Komprimierungsmethode fest. Das optionale Argument `flags` kann verwendet werden, um die Komprimierungsstufe zu definieren, wobei 1 die schnellste und 9 die höchste Komprimierung ist. Zulässige Werte sind 1-9 oder 0, um die Standardeinstellungen der Komprimierung zu verwenden.

Der Parameter `method` muss eine der folgenden Konstanten sein:

`#ZIP_CM_DEFAULT:`

Dies ist die Standardeinstellung. Derzeit das gleiche wie `#ZIP_CM_DEFLATE`.

`#ZIP_CM_STORE:`

Die Datei wird unkomprimiert gespeichert.

`#ZIP_CM_BZIP2:`

Die Datei wird mit dem bzip2-Algorithmus komprimiert.

`#ZIP_CM_DEFLATE:`

Die Datei wird mit dem ZLIB-Algorithmus und Standardoptionen komprimiert.

Beachten Sie, dass nur `#ZIP_CM_DEFLATE` und `#ZIP_CM_STORE` als universelle unterstützt gelten.

Beachten Sie auch, dass die Änderung des Zip-Archivs nicht sofort erfolgt, sondern verschoben wird, bis Sie `zip.CloseArchive()` aufrufen.

#### EINGABEN

<code>id</code>	Identifikator des zu verwendenden Zip-Archivs
<code>idx</code>	Index der Datei, deren Komprimierung eingestellt werden soll



<code>method</code>	Gewünschte Komprimierungsmethode (siehe oben)
<code>flags</code>	Optional: Gewünschter Komprimierungsgrad von 1 (am schnellsten) bis 9 (am höchsten) oder 0 für die Standardeinstellung der Komprimierung (standardmäßig 0)

## 4.16 `zip.SetFileEncryption`

### BEZEICHNUNG

`zip.SetFileEncryption` – setzt die Dateiverschlüsselung

### ÜBERSICHT

```
zip.SetFileEncryption(id, idx, method[, pwd$])
```

### BESCHREIBUNG

Dieser Befehl legt die Verschlüsselungsmethode für die Datei im Index `idx` in dem von `id` angegebenen Zip-Archiv fest. Die gewünschte Verschlüsselungsmethode muss im Parameter `method` übergeben werden. Optional können Sie im Argument `pwd$` ein Passwort angeben. Wenn das Argument `pwd$` weggelassen oder auf eine leere Zeichenkette gesetzt wird, wird das mit `zip.SetDefaultPassword()` festgelegte Standard-Passwort verwendet.

Der Parameter `method` muss eine der folgenden Konstanten sein:

<code>#ZIP_EM_NONE:</code>	Keine Verschlüsselung.
<code>#ZIP_EM_AES_128:</code>	Winzip AES-128 Verschlüsselung.
<code>#ZIP_EM_AES_192:</code>	Winzip AES-192 Verschlüsselung.
<code>#ZIP_EM_AES_256:</code>	Winzip AES-256 Verschlüsselung.

Beachten Sie, dass die Änderung am Zip-Archiv nicht sofort erfolgt, sondern verschoben wird, bis Sie `zip.CloseArchive()` aufrufen.

### EINGABEN

<code>id</code>	Identifikator des zu verwendenden Zip-Archivs
<code>idx</code>	Index der Datei, deren Verschlüsselung eingestellt werden soll
<code>method</code>	Gewünschte Verschlüsselungsmethode (siehe oben)
<code>pwd\$</code>	Optional: Gewünschtes Passwort für die Datei oder leere Zeichenkette, um das Standard-Passwort zu verwenden (voreingestellt ist eine leere Zeichenkette)

## 4.17 zip.SetFileTime

### BEZEICHNUNG

zip.SetFileTime – legt den Zeitstempel der Datei fest

### ÜBERSICHT

```
zip.SetFileTime(id, idx, time$)
```

### BESCHREIBUNG

Dieser Befehl legt den Datumsstempel für die Datei im Index `idx` in dem von `id` angegebenen Zip-Archiv fest. Der Datumsstempel muss im Standard-Hollywood-Datumsformat übergeben werden: `dd-mmm-yyyy hh:mm:ss`

Beachten Sie, dass die Änderung am Zip-Archiv nicht sofort erfolgt, sondern verschoben wird, bis Sie `zip.CloseArchive()` aufrufen.

### EINGABEN

<code>id</code>	Identifikator des zu verwendenden Zip-Archivs
<code>idx</code>	Index der Datei, deren Datumsstempel geändert werden soll
<code>time\$</code>	Gewünschter Datumstempel

## Anhang A Lizenzen

### A.1 LibZip Lizenz

Copyright (C) 1999-2016 Dieter Baron and Thomas Klausner

The authors can be contacted at <libzip@nih.at>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The names of the authors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### A.2 AES-Verschlüsselungsunterstützung

Copyright (c) 2002, Dr Brian Gladman, Worcester, UK. All rights reserved.

The free distribution and use of this software in both source and binary form is allowed (with or without changes) provided that:

1. distributions of this source code include the above copyright notice, this list of conditions and the following disclaimer;
2. distributions in binary form include the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other associated materials;
3. the copyright holder's name is not used to endorse products built using this software without specific written permission.

ALTERNATIVELY, provided that this notice is retained in full, this product may be distributed under the terms of the GNU General Public License (GPL), in which case the provisions of the GPL apply INSTEAD OF those given above.

DISCLAIMER: This software is provided 'as is' with no explicit or implied warranties in respect of its properties, including, but not limited to, correctness and/or fitness for purpose.

Issue Date: 18th November 2008



# Index

zip.AddDirectory .....	9	zip.LocateFile .....	16
zip.AddFile .....	9	zip.OpenArchive .....	17
zip.CloseArchive .....	11	zip.RenameFile .....	18
zip.DeleteFile .....	12	zip.SetDefaultPassword .....	19
zip.ExtractFile .....	12	zip.SetFileComment .....	19
zip.GetFileAtIndex .....	14	zip.SetFileCompression .....	20
zip.GetFileAttributes .....	14	zip.SetFileEncryption .....	21
zip.GetFileComment .....	15	zip.SetFileTime .....	21
zip.GetObjectType .....	16		

