HOCHSCHULE
MITTWEIDA
University of Applied Sciences

BPlist Datenformat

Beispieluntersuchung

Autor: Prof. Ronny Bodach

Plattformabhängige Speichermethode

Aufbau des Plist Formates

Die BPlist Datei besteht zunächst aus vier Bestandteilen:

- 8 Byte Header
- var. Byte Objekttabelle
- var. Byte Offset Table
- 32 Byte Trailer

Eine Binary Property List (bplist)-Datei besteht aus 4 Teilen. Sie werden im Folgenden nicht in der Reihenfolge beschrieben, in der sie auftreten, sondern in der Reihenfolge, die beim Lesen der Datei am sinnvollsten ist.

Plattformabhängige Speichermethode

Aufbau des Plist Formates - Header

Die bplist-Datei trägt im Header eine Signatur:

- **`0x62706C6973743030 = bplist`**,
- gefolgt von der **`Version`** des bplist Files (**`00,15,16,etc.`**).

Die Untersuchung von BPlist Dateien setzt in der Regel voraus, dass es sich um eine BPlist der Version 00 handelt, da andere Versionen von Apple bisher nicht dokumentiert sind.

Plattformabhängige Speichermethode

Aufbau des Plist Formates - Trailer

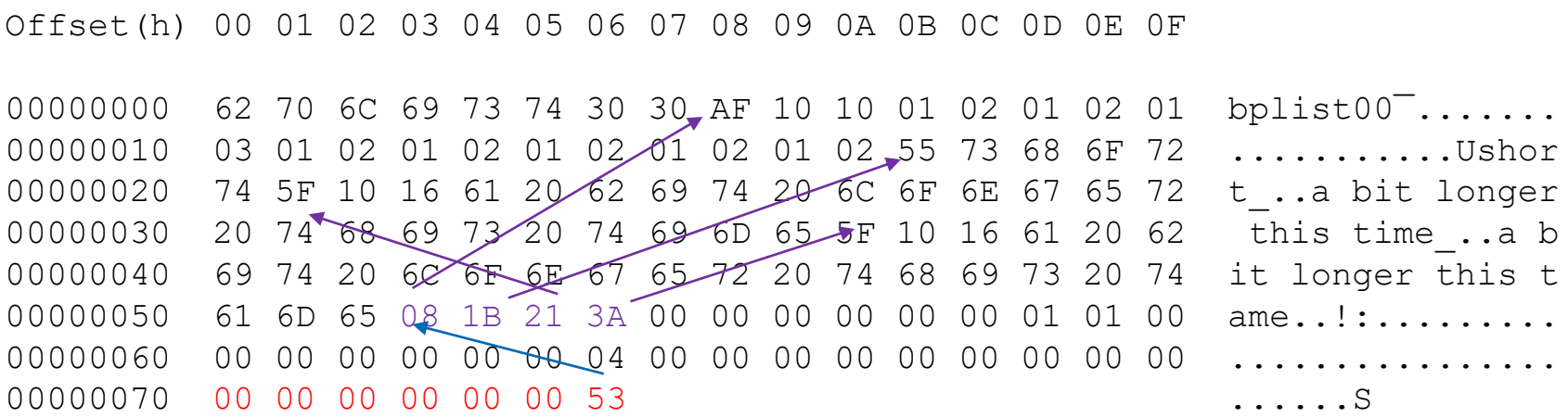
Der Trailer besteht im Wesentlichen nur aus drei, für uns relevanten Teilen. Zum einen die Offsetgröße in Byte, welche uns verrät, wie groß ein Offset in der Offset Tabelle ist. Weiter ist hier auch die Anzahl aller gespeicherten Objekte vermerkt. Abschließend ist der Offset der Offset Tabelle ebenfalls noch enthalten.

Offset	Länge	Datenstruktur	Beschreibung
0	5	unbenutzt	unbenutzt
6	1	8-bit unsigned integer	Offsetgröße in Byte
7	1	8-bit unsigned integer	Referenzgröße in Dictionaries
8	8	64-bit unsigned big endian integer	Anzahl der gespeicherten Objekte
16	8	64-bit unsigned big endian integer	Top Level Object Index
24	8	64-bit unsigned big endian integer	Offset der Offset Tabelle
32			

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Offset-Tabelle

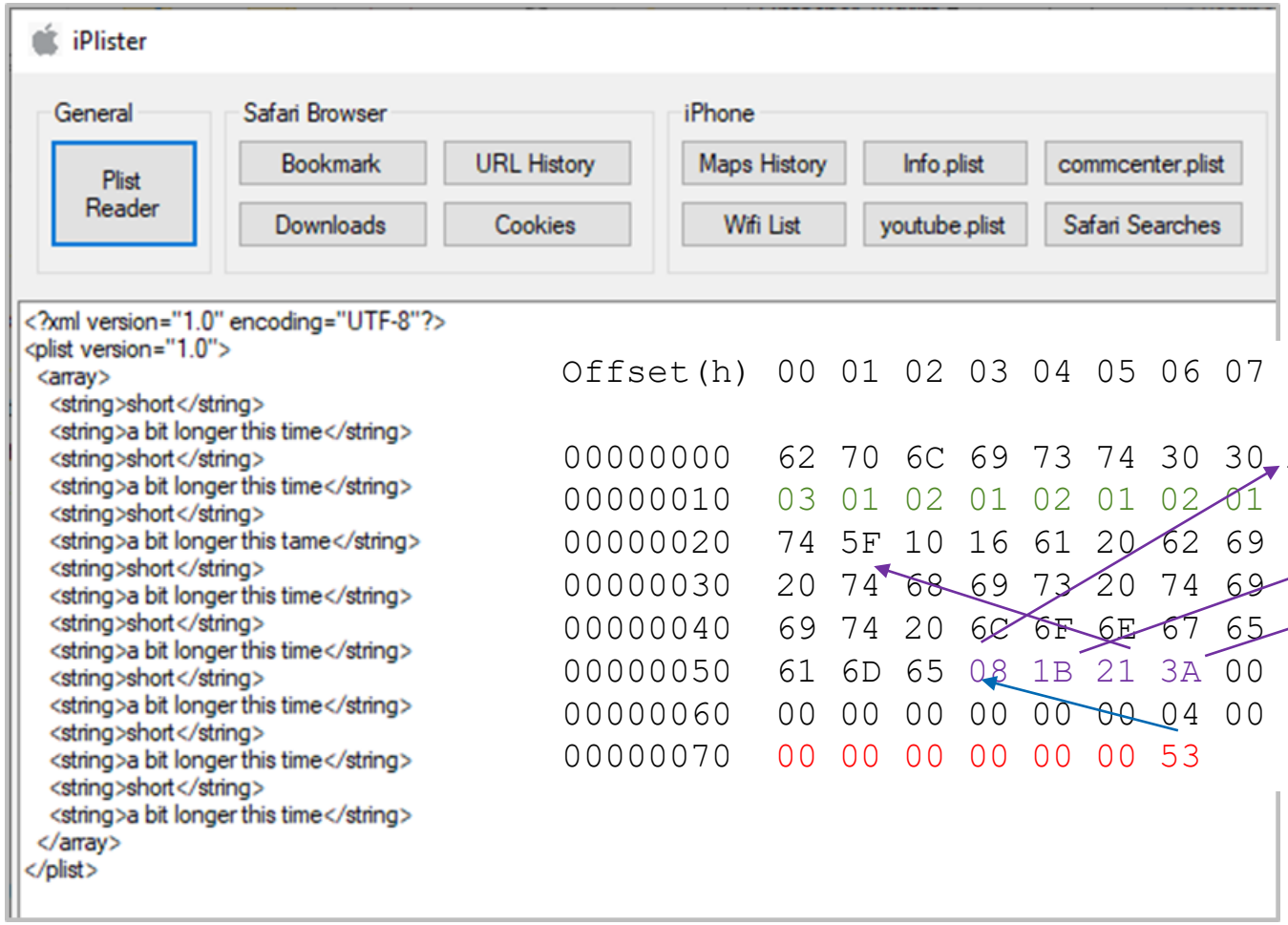
Die Offset-Tabelle ist der dritte Abschnitt in der Datei. Der Startoffset in der Datei wird im Trailer angegeben („Offset der Offset-Tabelle“). Die Länge kann durch Multiplizieren der Trailer-Felder „Anzahl der gespeicherten Objekte“ mit „Offsetgröße in Byte“ ermittelt werden. Die Offsettabelle enthält ein Array von Offsets mit einem Index von Null (dh der erste Eintrag wird als Eintrag 0 bezeichnet, der zweite als Eintrag 1 usw.), die auf Objekte in der Objekttabelle verweisen.



Beispiel Offset-Tabelle und referenzierte Objekte dazu

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Offset-Tabelle



Plattformabhängige Speichermethode

Aufbau des Plist Formates – die Objekttabelle

Die Objekttabelle ist der zweite Abschnitt in der Datei, der direkt nach dem Header bei Offset 8 beginnt und bis zum Beginn der Offset-Tabelle fortgesetzt wird. Es enthält die binären Darstellungen jedes „Objekts“ in der Plist.

Jedes Objekt besteht aus einem Typ-Deskriptor-Byte, optional gefolgt von einer Länge (abhängig vom Datentyp kann die Länge im Datentyp-Byte impliziert sein), gefolgt von den Daten selbst.

Um die Länge der Objekte festzustellen, muss zunächst der Typ erkannt werden. Dieser wird in den meisten Fällen durch die ersten 4 Bit bestimmt. Die Länge ergibt sich dann, durch die nächsten 4 Bit. Ist die Länge kleiner als 15 Byte so reichen diese 4 Bit aus um die Länge anzugeben. Ist das Objekt größer, so werden die 4 Bit alle mit 1 belegt. Das bedeutet, dass die nächsten Bytes für die Größe ausgewertet werden müssen.

Plattformabhängige Speichermethode

Aufbau des Plist Formates – die Objekttabelle

Typ	Binär	Größe	Inhalt
null	0000 0000		
bool	0000 1000		FALSE
bool	0000 1001		TRUE
fill	0000 1111		Füll Byte
int	0001 nnnn	…	2^nnnn Big-Endian Bytes
real	0010 nnnn	…	2^nnnn Big-Endian Bytes
date	0011 0011	…	8 Byte Float (Big-Endian)
data	0100 nnnn	[int] …	nnnn Bytes bis '1111', danach int Typ und int Count gefolgt von Inhalt
string	0101 nnnn	[int] …	ASCII String, nnnn Chars bis '1111', danach int Typ und int Count gefolgt von Inhalt
string	0110 nnnn	[int] …	Unicode String, nnnn Chars bis '1111', danach int Typ und int Count gefolgt von Inhalt

Plattformabhängige Speichermethode

Aufbau des Plist Formates – die Objekttabelle

Typ	Binär	Größe	Inhalt
	0111 xxxx		reserviert
uid	1000 nnnn	…	nnnn+1 Bytes
	1001 xxxx		reserviert
array	1010 nnnn	[int] objref*	nnnn Einträge bis ’1111’, danach int Typ und int Count
	1011 xxxx		reserviert
set	1100 nnnn	[int] objref*	nnnn Einträge bis ’1111’, danach int Typ und int Count
dict	1101 nnnn	[int] keyref	nnnn Einträge bis ’1111’, danach int Typ und int Count
	1110 xxxx		reserviert
	1111 xxxx		reserviert
	0111 xxxx		reserviert

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01  bplist00¯.......
00000010  03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72  ...........Ushor
00000020  74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72  t_..a bit longer
00000030  20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62   this time_..a b
00000040  69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74  it longer this t
00000050  61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00  ame..!:.........
00000060  00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 53                             ......S
```

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01  bplist00¯.......
00000010  03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72  ...........Ushor
00000020  74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72  t_..a bit longer
00000030  20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62   this time_..a b
00000040  69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74  it longer this t
00000050  61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00  ame..!:.........
00000060  00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 53                             ......S
```

AF = 1010 1111

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01  bplist00¯.......
00000010  03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72  ...........Ushor
00000020  74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72  t_..a bit longer
00000030  20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62   this time_..a b
00000040  69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74  it longer this t
00000050  61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00  ame..!:.........
00000060  00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 53                             ......S
```

AF = 1010 1111

array	1010 nnnn	[int] objref*	nnnn Einträge bis ’1111’, danach int Typ und int Count

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01  bplist00¯.......
00000010   03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72  ...........Ushor
00000020   74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72  t_..a bit longer
00000030   20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62   this time_..a b
00000040   69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74  it longer this t
00000050   61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00  ame..!:.........
00000060   00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00  ................
00000070   00 00 00 00 00 00 53                             ......S
```

AF = 1010 1111

array	1010 nnnn	[int] objref*	nnnn Einträge bis '1111', danach int Typ und int Count

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01  bplist00¯.......
00000010   03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72  ...........Ushor
00000020   74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72  t_..a bit longer
00000030   20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62   this time_..a b
00000040   69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74  it longer this t
00000050   61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00  ame..!:.........
00000060   00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00  ................
00000070   00 00 00 00 00 00 53                             ......S
```

AF = 1010 1111

nnnn = Anzahl im nachfolgenden Byte

array	1010 nnnn	[int] objref*	nnnn Einträge bis '1111', danach int Typ und int Count

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01  bplist00¯.......
00000010   03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72  ...........Ushor
00000020   74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72  t_..a bit longer
00000030   20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62   this time_..a b
00000040   69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74  it longer this t
00000050   61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00  ame..!:.........
00000060   00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00  ................
00000070   00 00 00 00 00 00 53                             ......S
```

AF = 1010 1111

nnnn = Anzahl im nachfolgenden Byte

array	1010 nnnn	[int] objref*	nnnn Einträge bis '1111', danach int Typ und int Count

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01  bplist00¯.......
00000010   03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72  ...........Ushor
00000020   74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72  t_..a bit longer
00000030   20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62   this time_..a b
00000040   69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74  it longer this t
00000050   61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00  ame..!:.........
00000060   00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00  ................
00000070   00 00 00 00 00 00 53                             ......S
```

10 Type = 0001 0000

2^0 = 1 Byte

int	0001 nnnn	...	2^nnnn Big-Endian Bytes

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01  bplist00¯.......
00000010   03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72  ...........Ushor
00000020   74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72  t_..a bit longer
00000030   20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62   this time_..a b
00000040   69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74  it longer this t
00000050   61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00  ame..!:.........
00000060   00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00  ................
00000070   00 00 00 00 00 00 53                             ......S
```

10 Type = 0001 0000

10 Count = 16 Dezimal

$2^{\wedge 0}$ = 1 Byte

int	0001 nnnn	...	$2^{\wedge nnnn}$ Big-Endian Bytes

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01  bplist00¯.......
00000010   03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72  ...........Ushor
00000020   74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72  t_..a bit longer
00000030   20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62   this time_..a b
00000040   69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74  it longer this t
00000050   61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00  ame..!:.........
00000060   00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00  ................
00000070   00 00 00 00 00 00 53                             ......S
```

10 Count = 16 Dezimal

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01  bplist00¯.......
00000010   03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72  ...........Ushor
00000020   74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72  t_..a bit longer
00000030   20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62   this time_..a b
00000040   69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74  it longer this t
00000050   61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00  ame..!:.........
00000060   00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00  ................
00000070   00 00 00 00 00 00 53                             ......S
```

10 Count = 16 Dezimal

16 Array Member – Nummer bezieht sich auf Position innerhalb der Plist Offsets (beginnend bei 0):

01 02 01 02 01 03 01 02 01 02 01 02 01 02 01 02

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01   bplist00¯.......
00000010   03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72   ...........Ushor
00000020   74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72   t_..a bit longer
00000030   20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62    this time_..a b
00000040   69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74   it longer this t
00000050   61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00   ame..!:.........
00000060   00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00   ................
00000070   00 00 00 00 00 00 53                              ......S
```

16 Array Member – Nummer bezieht sich auf Position innerhalb der Plist Offsets (beginnend bei 0):

01 02 01 02 01 03 01 02 01 02 01 02 01 02 01 02

Plattformabhängige Speichermethode

Aufbau des Plist Formates – Beispiel

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   62 70 6C 69 73 74 30 30 AF 10 10 01 02 01 02 01   bplist00¯.......
00000010   03 01 02 01 02 01 02 01 02 01 02 55 73 68 6F 72   ...........Ushor
00000020   74 5F 10 16 61 20 62 69 74 20 6C 6F 6E 67 65 72   t_..a bit longer
00000030   20 74 68 69 73 20 74 69 6D 65 5F 10 16 61 20 62    this time_..a b
00000040   69 74 20 6C 6F 6E 67 65 72 20 74 68 69 73 20 74   it longer this t
00000050   61 6D 65 08 1B 21 3A 00 00 00 00 00 00 01 01 00   ame..!:.........
00000060   00 00 00 00 00 00 04 00 00 00 00 00 00 00 00 00   ................
00000070   00 00 00 00 00 00 53                              ......S
```

```
<plist version="1.0">
<array>
    <string>short</string>
    <string>a bit longer this time</string>
    <string>short</string>
    <string>a bit longer this time</string>
    <string>short</string>
    <string>a bit longer this tame</string>
    <string>short</string>
    <string>a bit longer this time</string>
    <string>short</string>
    <string>a bit longer this time</string>
    <string>short</string>
    <string>a bit longer this time</string>
    <string>short</string>
    <string>a bit longer this time</string>
    <string>short</string>
    <string>a bit longer this time</string>
</array>
</plist>
```

16 Array Member – Nummer bezieht sich auf Position innerhalb der Plist Offsets (beginnend bei 0):

01 02 01 02 01 03 01 02 01 02 01 02 01 02 01 02