New Technical Notes

Macintosh

Developer Support

Version Territory

Overview

Revised by: Rich Collyer October 1990
Written by: Darin Adler April 1988

This Technical Note describes the `'vers'` resource supported by Finder 6.1 and later.
Changes since April 1989: Changed MPW C code to reflect the changes in MPW C 3.1.

Finder 6.1 introduced a feature allowing the creator of a file to identify the version of that file as well as the version of a set of files which includes that file. These version numbers are stored in `'vers'` resources, and each contains a BCD form of the version number and a longer version message (which the Finder displays in the Get Info window for each file).

Apple's Version Numbering Scheme

Apple uses a version numbering scheme for its software products which you might want to adopt. Table 1 summarizes the scheme, which involves three numbers, each separated by periods.

Event	Version
First released version	1.0
First revision	1.1
First bug fix to the first revision	1.1.1
First major revision or rewrite	2.0

Table 1–Apple's Version Numbering Scheme

Note that Apple increments the first number when it releases a major revision, the second number when it releases a minor revision, and the third number when it releases a version to address bugs (the third number is omitted if it is zero).

During product development, Apple uses a version number followed by a suffix which indicates the stage of development. Table 2 presents a few examples.

Event	Version	Stage
First versions	1.0d1, 1.0d2…	development
Product features defined (begin testing)	1.0a1, 1.0a2…	alpha
Product is stable (begin final testing)	1.0b1, 1.0b2…	beta
First released version	1.0	release
First revision	1.1d1, 1.1d2, 1.1d3…1.1	
First bug fix to the first revision	1.1.1d1, 1.1.1d2, 1.1.1d3…1.1.1	
First major revision	2.0d1, 2.0d2, 2.0d3 2.0	

Table 2–Development Version Numbering

Version Resources

Each `'vers'` resource has the following format (described with a Rez template):

```
#include "SysTypes.r" /* for country codes */

type 'vers' {
    byte;             /* first part of version number in BCD */
    byte;             /* second and third parts of version number */
    byte development=0x20, alpha=0x40, beta=0x60, release=0x80;
    byte;             /* stage of non-release version */
    integer Country;  /* country code as in international utilities */
    pstring;          /* short version number */
    pstring;          /* long version message */
};
```

The short version number is a string which only contains the version number (e.g., “1.0”). The long version message can also include a copyright notice, a release date, or other information, but it should **not** include the name of the program. The following examples illustrate the proper use of the Rez template to create version resources:

```
resource 'vers' (1) {
    0x01, 0x00, release, 0x00, verUS,
    "1.0",
    "1.0 (US), ©1989 Inside Joke"
};
resource 'vers' (2) {
    0x12, 0x00, release, 0x00, verUS,
    "12.0",
    "Watt-R-Utilities Disk 12.0"
};

resource 'vers' (1) {
    0x23, 0x45, beta, 0x67, verFinland,
    "23.4.5b67",
    "23.4.5b67 (Finland), ©1989 Squid, Inc."
};
```

```
resource 'vers' (2) {
    0x55, 0x00, development, 0x67, verFinland,
    "55.0d67",
    "Friends of Skippy White 55.0d67"
};
```

The following is a type definition for `'vers'` resources in MPW Pascal:

```
NumVersion = PACKED RECORD
    CASE INTEGER OF
      0:
        (majorRev: SignedByte;  {1st part of version number in BCD}
        minorRev: 0..9;         {2nd part is 1 nibble in BCD}
        bugFixRev: 0..9;        {3rd part is 1 nibble in BCD}
        stage: SignedByte;      {stage code: dev, alpha, beta, final}
        nonRelRev: SignedByte); {revision level of non-released version}
      1:
        (version: LONGINT);     {to use all 4 fields at one time}
    END;

{ Numeric version part of 'vers' resource }
VersRecPtr = ^VersRec;
VersRecHndl = ^VersRecPtr;
VersRec = RECORD
    numericVersion: NumVersion; {encoded version number}
    countryCode: INTEGER;       {country code from intl utilities}
    shortVersion: Str255;       {version number string - worst case}
    reserved: Str255;           {longMessage string packed after shortVersion}
    END;
```

The type definition in MPW C is as follows (these structures are not needed in your code since they are included in the header file Files.h):

```
struct NumVersion {
    unsigned char majorRev;     /*1st part of version number in BCD*/
    unsigned int minorRev : 4;  /*2nd part is 1 nibble in BCD*/
    unsigned int bugFixRev : 4; /*3rd part is 1 nibble in BCD*/
    unsigned char stage;        /*stage code: dev, alpha, beta, final*/
    unsigned char nonRelRev;    /*revision level of non-released version*/
};

/* Numeric version part of 'vers' resource */
struct VersRec {
    NumVersion numericVersion;  /*encoded version number*/
    short countryCode;          /*country code from intl utilities*/
    Str255 shortVersion;        /*version number string - worst case*/
    Str255 reserved;            /*longMessage string packed after shortVersion*/
};

typedef VersRec *VersRecPtr, **VersRecHndl;
```

The `longMessage` string is not necessarily word-aligned due to the way the resource is formatted, so you should use `_BlockMove` to extract it from the record. Following are examples of this technique:

MPW Pascal

```
VAR
    version: VersRecHandle;
    messagePtr: StringPtr;
    longMessage: Str255;

version := GetResource ('vers', 1);
WITH version^^ DO
    BEGIN
    {calculate a pointer to the long message}
    messagePtr := StringPtr(Ord(@shortVersion[1])+Length(shortVersion));
    {move the long message into a string}
    BlockMove(Ptr(messagePtr), @longMessage, Length(messagePtr^)+1);
    END;
```

MPW C

```
VersRecHndl  version;
StringPtr    messagePtr;
char         *shortversion, *longversion;

version = (VersRecHndl) GetResource ('vers', 1);
/* calculate a pointer to the long message */
messagePtr = (StringPtr) (((unsigned long) &(**version).shortVersion[1]) +
             ((**version).shortVersion[0]));
/* move the long message into a string */
BlockMove(messagePtr, &longMessage, ((unsigned char) &messagePtr) + 1);
```

A file can contain either one, two, or no `'vers'` resources. If present, a `'vers' (1)` resource identifies the file version while a `'vers' (2)` resource identifies the version (and name) of a set of files which includes that file, thus linking all the files which make up the set. Apple uses this mechanism to identify System Software versions. All files on System Tools disks have a `'vers' (2)` resource that identifies the version of System Tools with which they were released. In addition, each file has a `'vers' (1)` resource that identifies the version of the particular file.

Version Resources and the Finder

The Finder displays the long message from `'vers' (1)` and `'vers' (2)` resources, if they are present, in the Get Info window of a file; it ignores the rest of the `'vers'` resource. Following is an example of the `'vers'` resources from Finder 6.1 with a Get Info window for the Finder file in Figure 1.

```
resource 'vers' (1) {
    0x06, 0x10, release, 0x00, verUS,
    "6.1",
    "6.1, Copyright Apple Computer, Inc. 1983-88"
};

resource 'vers' (2) {
    0x06, 0x03, release, 0x00, verUS,
    "6.0.3",
    "System Software Version 6.0.3"
};
```

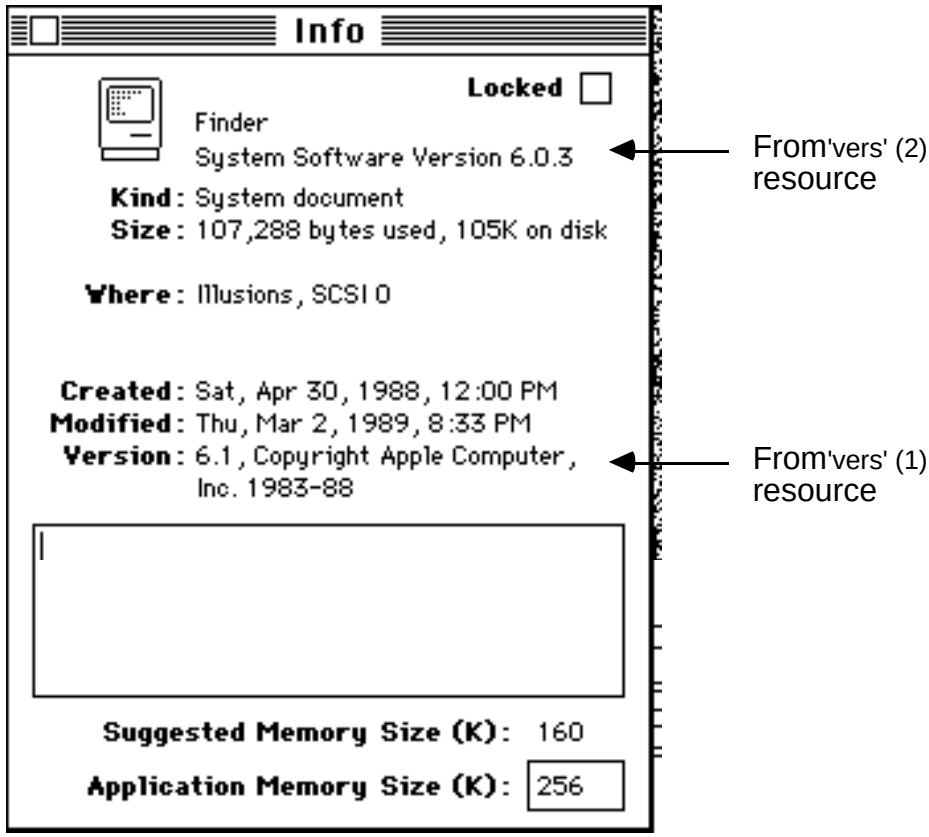

Figure 1–Get Info Window for the Finder File

The other fields (besides the long message) are often useful to applications other than the Finder. The short version number is good for displaying the version of a particular file, as the Finder does for the System and Finder in the About the Macintosh Finder window. The BCD version number is well suited for checking for a desired version number or comparing two versions. Note that this BCD numbering scheme represents a more recent version with a number greater than an older version, so a numeric comparison between two four-byte values is all that is necessary to determine which value is the most recent.

Final Note

The Finder Interface chapter of *Inside Macintosh*, Volume III-7 describes a resource (part of the bundle) that contains the **version data** of an application. This version data is typically a string that gives the name, version number, and date of the application. The Finder displays the version data (treating it as a string) in the Get Info window if there is no `'vers' (1)` resource in the application. Unlike this version data in an application, any type of file can contain `'vers'` resources, not just those files which contain bundles.

Further Reference:

- *Inside Macintosh*, Volume III-7, The Finder Interface
- M.TB.Bundles