

Voxel File Format Specification

August 2001



Part Number: 81-0009

Copyright and Disclaimer

TeraRecon, Inc.
300 Baker Avenue, Suite 301
Concord, MA 01742
www.terarecon.com

TeraRecon, Inc. makes no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description. Possession, use, or copying of the software described in this publication is authorized only pursuant to a valid written license from TeraRecon, Inc. or an authorized sublicensor.

©TeraRecon, Inc. 2001. All rights reserved. Printed in U.S.A.

The following are trademarks of TeraRecon, Inc.: VolumePro, VolumePro 500, VolumePro 1000, VolumePro Net, Volume Library Interface, vg500, vg1000, Real Time Visualization, and RTViz.

The following are third-party trademarks: Adobe, Acrobat, and Acrobat Exchange are trademarks of Adobe Systems Incorporated. Windows NT and Internet Explorer are trademarks and Windows, Windows 95, Windows 2000, and MS-DOS are registered trademarks of Microsoft Corporation. Netscape and Netscape Navigator are registered trademarks of Netscape Communications Corporation. Pentium is a registered trademark of Intel Corporation. OpenGL and IRIX are trademarks of Silicon Graphics, Inc. Sun, Sun Microsystems, Java, Solaris, and Ultra are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. 3Dlabs and Oxygen are trademarks or registered trademarks of 3Dlabs Ltd., 3Dlabs Inc. Ltd., or 3Dlabs Inc. in the United States and other countries.

Table of Contents

Voxel File Format Specification	1-1
Introduction.....	1-3
File Structure Overview	1-3
File Header.....	1-4
Signature - Required - Singular	1-4
Comments - Optional - Multiple.....	1-5
Volume Count - Optional - Singular.....	1-5
Title - Optional - Multiple	1-5
Copyright Notice - Optional - Multiple	1-5
Attribute - Optional - Multiple.....	1-5
Data - Optional - Multiple	1-6
End of Section - Required - Singular	1-6
Volume Description.....	1-6
Start of Section - Required - Singular.....	1-6
Volume Size - Required - Singular.....	1-6
Voxel Size - Required - Singular	1-7
Endian-ness - Required - Singular	1-7
Volume Scale - Optional - Singular.....	1-7
Volume Position - Optional - Singular	1-7
Voxel Field - Field 0 Required, Others Optional - Multiple	1-8
Model Matrix - Optional - Singular	1-10
Attribute - Optional - Multiple.....	1-11
Additional Data - Optional - Multiple	1-11
Comments - Optional - Multiple.....	1-11
Title - Optional - Multiple	1-11
Copyright - Optional - Multiple.....	1-11
End of Section - Required - Singular	1-11
Examples.....	1-12
Volume Data.....	1-12
Voxel Data	1-12
Additional Data.....	1-12
End of Data	1-12
 Appendix A: Recognized Names	 A-1
 Appendix B: Revision History	 B-1

Voxel File Format Specification

Topics described in this document include:

Introduction	page 1-3
File Structure Overview	page 1-3
File Header	page 1-4
Signature - Required - Singular	page 1-4
Comments - Optional - Multiple	page 1-5
Volume Count - Optional - Singular	page 1-5
Title - Optional - Multiple	page 1-5
Copyright Notice - Optional - Multiple	page 1-5
Attribute - Optional - Multiple	page 1-5
Data - Optional - Multiple	page 1-6
End of Section - Required - Singular	page 1-6
Volume Description	page 1-6
Start of Section - Required - Singular	page 1-6
Volume Size - Required - Singular	page 1-6
Voxel Size - Required - Singular	page 1-7
Endian-ness - Required - Singular	page 1-7
Volume Scale - Optional - Singular	page 1-7
Volume Position - Optional - Singular	page 1-7
Voxel Field - Field 0 Required, Others Optional - Multiple	page 1-8
Model Matrix - Optional - Singular	page 1-10
Attribute - Optional - Multiple	page 1-11
Additional Data - Optional - Multiple	page 1-11
Comments - Optional - Multiple	page 1-11
Title - Optional - Multiple	page 1-11
Copyright - Optional - Multiple	page 1-11
End of Section - Required - Singular	page 1-11
Examples	page 1-12

Volume Data	page 1-12
Voxel Data	page 1-12
Additional Data	page 1-12
End of Data	page 1-12

Introduction

This document describes the **vox1999a** file format; a simple and flexible format for voxel data storage, exchange, and input and output filter targeting. One of the design goals is the minimum required effort to read and render voxel data stored in such a format by TeraRecon's VolumePro family of volume rendering hardware, while not making the file format specific to the VolumePro.

The proposed file format is operating system and processor independent. It is capable of storing 1, 8, 16, 32, and 64 bit volumes, each of which can contain multiple fields. This format additionally enables you to store multiple volumes in a single file.

Volumes stored in this format are assumed to be *parallelepiped* (i.e., consisting of slices of voxels with equal distance between the slices).

Although it is understood that compression is useful in improving disk space utilization and reducing file loading and transmission time, as of this version, no compression is proposed.

File Structure Overview

The proposed voxel file format consists of three logical sections. The file begins with a *file header* that identifies the file as being the proposed voxel file format and contains descriptions and copyright notices pertaining to the data in the file. The header is followed by a *volume description* section that describes the first volume and the voxel data in it. The *volume data* section follows immediately after the end of the description section. Each datum in the data section represents a voxel containing one or more fields (for example, an RGBA voxel). You may repeat the volume description and volume data sections to provide for multiple volumes in one file.

The file structure is depicted as follows:

File Structure

Section	Description
File Header	(signature, number of volumes, copyrights, descriptions)
Volume Description	(volume dimension, voxel size, voxel fields)
Volume Data	(voxel data streams, always ends at a byte boundary)
Volume Description	
Volume Data	
...	

The file header and the volume description sections contain text in ASCII encoding, organized as multiple name-value pairs called *descriptors*. Each descriptor must be on a separate line, and the descriptor must be complete on that line (except for the two descriptors that accept multiple fields delimited by parentheses). The name is case-sensitive. Leading spaces before the name are allowed, with a few exceptions noted. There must be one or more spaces between the name and value.

In the following definitions of descriptors, we use the indicated parameter types:

Parameter Types Used in Descriptors

Name	Description
int	An integer value. Currently no negative values are useful.
float	A floating point value, in a form suitable for “atof”.
word	Either a sequence of non-blank characters (which must not start with a double-quote (ASCII value 0x22)) or a <code>quoted-string</code> as described below.
rest-of-line	A sequence of characters terminated by the end of line.
quoted-string	A sequence of characters starting and ending with a double-quote (ASCII value 0x22). This sequence must be complete on one line, and may contain the character sequence backslash double-quote (ASCII values 0x5C 0x22), which represents the single character double-quote.

The end of line indicator is newline, ASCII value 0x0A. Blank characters can be blanks (ASCII value 0x20) or tabs (ASCII value 0x04), but not newlines, except as noted for specific descriptors.

All `Vox1999a` file readers must accept all descriptors described in this document without error. (A warning may be issued for non-processed descriptors, but the file reader must be able to continue processing the file.)

File Header

The voxel data file starts with a file header that identifies the file as being in the `Vox1999a` format and describes the content of the file. The file header ends when the end-of-section marker is encountered.

Signature - Required - Singular

`Vox1999a`

The signature must appear at the beginning of the file (bytes 0-7). This signature must be immediately followed by the end of line.

Comments - Optional - Multiple

```
// rest-of-line
```

Comments are introduced by `//` (ASCII values 0x2F 0x2F) and terminated by the end of line. The double slashes *must* be the first two characters on the line; you cannot have a comment following a descriptor.

Comments can appear anywhere in the file header after the signature.

Volume Count - Optional - Singular

```
VolumeCount int
```

This descriptor specifies the number of volumes stored in the file. If the `int` is 0, or this descriptor is not present, an arbitrary number of volumes are stored in the file. The last volume is followed by the end of file.

Title - Optional - Multiple

```
Title rest-of-line
```

Use this descriptor to hold the descriptions of the overall data, such as the origin of the data, date it was acquired/generated, etc. This descriptor ends at a new line.

You can use multiple title descriptors in the file header as in the following example:

```
Title CT scan at XY hospital on 2000/01/21
Title 120kV, 300mA, 40DFOV
```

Copyright Notice - Optional - Multiple

```
Copyright rest-of-line
```

Use this descriptor to place a copyright notice. Similar to the title descriptor, multiple copyright notice descriptors can be present in the file header.

Attribute - Optional - Multiple

```
Attribute word rest-of-line
```

This descriptor is available for application specific purposes, for textual (ASCII) data related to the entire file. All `Vox1999a` file readers should store all attribute word/string pairs for application retrieval.

Data - Optional - Multiple

Data word int

This descriptor is available for application specific purposes, for binary data related to the entire file. The integer argument represents the size, in bytes, of the data, which must appear after the end-of-section marker. All `Vox1999a` file readers must process all `Data` descriptors to allow proper computation of file offsets for subsequent file sections. All `Vox1999a` file readers should store all data word/size/binary-data triplets for application retrieval.

End of Section - Required - Singular

##\f

This sequence signals the end of the file header. It must appear on a single line with no leading spaces. This sequence must be immediately followed by the end of line. “\f” denotes a form feed (ASCII value 0x0C). Application specific binary data blocks follow immediately after the end of this line, in the order that the `Data` descriptors appear in the file header, and with no padding.

Volume Description

The volume description section starts with a start-of-section marker and terminates with an end-of-section marker. Use the volume description section to describe the characteristics of the volume and the voxels. *Singular* and *multiple* refer to a single volume description, not to the entire `Vox1999a` file.

Start of Section - Required - Singular

##

This sequence signals the start of the volume description. It must appear on a single line with no leading spaces. This sequence must be immediately followed by the end of line. It must also follow immediately after the end-of-section marker, if there is no application specific binary data, or follow immediately after that data, if it is present.

Volume Size - Required - Singular

VolumeSize int int int

This descriptor describes the size of the volume. The three arguments are the number of voxels in the x , y , and z dimensions of the volume. Voxels are stored in a slice-by-slice fashion along the z -axis, with the index x running the fastest.

Voxel Size - Required - Singular

VoxelSize int

This descriptor specifies the size of voxel data stored in the volume data section immediately following the descriptor block. The value is the number of bits per voxel. Valid values are 1, 8, 16, 32, and 64. All `Vox1999a` file readers must be able to process all of these sizes sufficiently to allow proper computation of file offsets for subsequent file sections. A `Vox1999a` file reader is not required to fully support all sizes, particularly sizes 1 and 64.

Endian-ness - Required - Singular

Endian word

This descriptor specifies the *endianess* of the voxel data. The argument must be either `L` or `B` to indicate little-endian and big-endian modes, respectively. All `Vox1999a` file readers must be able to convert the voxel data into the endian mode supported by the system on which the reader is being run.

Volume Scale - Optional - Singular

VolumeScale float float float

This descriptor describes the scale of the three axis. The three floating point numbers represent the voxel spacing between adjacent voxels in each direction.

Note that this descriptor is for calibration information associated with the volume data and, as such, is not intended to be used in rendering, for which the model matrix should be used.

Volume Position - Optional - Singular

VolumePosition float float float

This descriptor describes the position of the volume. The three floating point numbers are the position of the voxel at index $(0,0,0)$, relative to some calibration point or application specified origin (the landmark for a CT scan, for example).

Given the scale of the axes and the position of the volume, the location of a voxel at (x, y, z) relative to the calibration point/application specific origin is:

$$\text{Location} = (x*S_x + P_x, y*S_y + P_y, z*S_z + P_z)$$

where S_x , S_y , and S_z are the arguments to `VolumeScale` and P_x , P_y , and P_z are the arguments to `VolumePosition`.

Like the `VolumeScale` descriptor, this descriptor is not intended to be used for rendering. To position the volume in a some graphical *World Space*, the model matrix should be used.

Voxel Field - Field 0 Required, Others Optional - Multiple

```
Field int(Position int
          Size int
          Name word
          [Format word]
          [Offset float]
          [Scale float]
          [Description quoted-string]
        )
```

The first line of this descriptor must include `Field` and the field number (the first argument). The name-value pairs (“field specifiers”) inside the parentheses can appear in any order. The end of line indicator can appear anywhere that blank space is acceptable. The close parenthesis must be followed by the end of line indicator, possibly after some blank space. None of the field specifiers may appear more than once in a field descriptor.

Field specifiers have the following characteristics:

Characteristics of Field Specifiers

Name	Required	Argument Type	Default Value	Description
Position	yes	int	N/A	Bit position in the voxel; 0 is the least significant bit
Size	yes	int	N/A	Number of bits in the field
Name	yes	word	N/A	Single word name for this field
Format	no	word	u	Field data format; see table on page 1-9
Offset	no	float	0.0	Offset to add to the field to get an application specific value. Note that this does not affect the voxel data read from the file and presented to the application.
Scale	no	float	1.0	Scale factor by which to multiply the field to get an application specific value. Note that this does not affect the voxel data read from the file and presented to the application.
Description	no	quoted-string	none	Application specific textual description of the field

For example, for a 32 bit RGBA volume with red as the least significant byte and alpha as the most significant byte, you can specify the following to describe the voxels in the volume:

```
Field 0 (Position 0 Size 8 Name Red)
Field 1 (Position 8 Size 8 Name Green)
Field 2 (Position 16 Size 8 Name Blue)
Field 3 (Position 24 Size 8 Name Alpha)
```

In general, this file format does not attach any meaning to the voxel fields, which can be anything (gradients, distance maps, segmentation information, etc.). Only `Field 0` must be specified. For example, if 12 bit CT scan data are stored in the upper 12 bits of a 16 bit quantity, the following field descriptor is sufficient:

```
Field 0 (Position 4 Size 12 Name CT_scan)
```

The `Format` specifier determines how the field should be interpreted. The standard format names and their meanings are shown in the table below. The default format is `u`, and all `Vox1999a` file readers must support this format.

Standard Field Format Names

Name	Description	Value Range	Representation
<code>u</code> or <code>uf</code>	unsigned fraction	0.0 to 1.0	0 to $(2^{\text{Size}} - 1)$
<code>sf</code>	signed fraction	-1.0 to 1.0	Sign/magnitude format. The sign is in the high order bit; magnitude is in the range 0 to $(2^{\text{Size}-1} - 1)$.
<code>ui</code>	unsigned integer	0 to $(2^{\text{Size}} - 1)$	0 to $(2^{\text{Size}} - 1)$
<code>si</code>	signed integer	$-(2^{\text{Size}-1} - 1)$ to $(2^{\text{Size}-1} - 1)$	Sign/magnitude format. The sign is in the high order bit; magnitude is in the range 0 to $(2^{\text{Size}-1} - 1)$.
<code>f</code>	floating point	IEEE single precision floating point range	IEEE single precision: must be 32 bits in size

A `Vox1999a` file reader may support additional format names, but to avoid conflict with future versions of this specification it should use all upper case names.

Two optional field specifiers (`Offset` and `Scale`) are available to convert the field value to the physical value it represents. You should use the `Offset` and `Scale` field specifiers as follows:

```
PhysicalValue = Offset + Scale * fieldValue
```

`Offset` and `Scale` should be assumed to be 0.0 and 1.0 if they are not present in the field descriptor. These specifiers must not affect the value of the field that a `Vox1999a` file reader presents to the application; they are intended solely for application usage.

For example, whereas the values for a 12 bit unsigned integer field can range from 0 to 4095, the CT numbers they represent are actually from -1024 to about 3071. With `Offset`, the field can be specified as follows:

```
Field 0 (Position 4 Size 12 Name CT_Data Offset -1024)
```

This allows the application to adjust the voxel field values to the proper physical values for any purpose it might have (for example, for presentation to the user).

All `Vox1999a` file readers must accept all the field specifiers described. All `Vox1999a` file readers must be able to present the `Position`, `Size`, and `Name` specifiers to the application; they should be able to present *all* field specifiers to the application.

Model Matrix - Optional - Singular

```
ModelMatrix (f11 f21 f31 f41  
             f12 f22 f32 f42  
             f13 f23 f33 f43  
             f14 f24 f34 f44)
```

This descriptor specifies a 4x4 model matrix to describe how the voxel center positions map to a *corrected* coordinate space that is rectilinear and scaled equally in all three dimensions. If this descriptor is not present, it is assumed that the model matrix is the identity matrix.

The model matrix is arranged in column major order (the order used in OpenGL and VLI). The translation terms are `f14`, `f24`, and `f34`; they may contain values to position the center of interest in the volume data set to the origin of the corrected coordinate space.

You can separate the matrix values by arbitrary white space (blanks, tabs, and end of line indicators), or by a single comma with optional white space before and after it.

All `Vox1999a` file readers must be able to present the model matrix to the application.

Attribute - Optional - Multiple

```
Attribute word rest-of-line
```

This descriptor is available for application specific purposes, for textual (ASCII) data related to this volume. All V_{OX1999a} file readers should store all attribute word/string pairs for application retrieval.

Additional Data - Optional - Multiple

```
Data word int
```

This descriptor specifies additional data that is to follow the voxel data. `word` is an application specified name for the data and `n` is the total size, in bytes, of the data. More than one data descriptor may appear in the descriptor, and the total size of the additional data following the voxel data is the sum of each individual data size. All V_{OX1999a} file readers must be able to skip this data so the presence of it does not affect multiple volume reading. All V_{OX1999a} file readers should store all data word/size/binary-data triplets for application retrieval.

Comments - Optional - Multiple

```
// rest-of-line
```

Comments are introduced with `//` (ASCII values 0x2F 0x2F) and terminate at the end of line. Comments can appear anywhere in the volume description section after the start-of-section marker. The double slashes *must* be the first two characters on the line; you cannot have a comment following a descriptor.

Title - Optional - Multiple

```
Title rest-of-line
```

You can use this descriptor to hold a description of the volume, in addition to the `TitLe` descriptors in the file header. Multiple `TitLe` descriptors can be present in the volume description.

Copyright - Optional - Multiple

```
Copyright rest-of-line
```

Additional copyright notice for the volume data. Multiple copyright notices can appear in the volume description.

End of Section - Required - Singular

```
##\f
```

This sequence signals the end of the volume description section. It must appear on a single line with no leading spaces. This sequence must be immediately followed by the end of line. “`\f`” denotes a form feed (ASCII value 0x0C).

Examples

An example of the simplest file header and volume description for a volume of single field voxels:

```
Vox1999a
##\f
##
VolumeSize 128 128 128
VoxelSize 16
Endian B
Field 0 (Position 0 Size 16 Name Test_Data)
##\f
```

An example of multiple field voxels:

```
Vox1999a
##\f
##
VolumeSize 256 256 300
VoxelSize 32
Endian B
Field 0 (Position 0 Size 16 Name segment)
Field 1 (Position 16 Size 16 Name MRIData)
// a volume consisting of raw MRI data (16bit) and 16 bits of
// segmentation information
##\f
```

Volume Data

Voxel Data

Voxel data are always packed with no padding, except at the end of the volume where the data stream is padded so it ends on a byte boundary. With the information in the data descriptor block, the total size, in bytes, of the volume can be computed as follows:

$$\text{TotalBytes} = \text{Floor}((\text{Nx} * \text{Ny} * \text{Nz} * \text{VoxelSize} + 7) / 8)$$

Additional Data

Additional data, if any, follows immediately after the voxel data. The total size of the additional data is the sum of the sizes of all Data descriptors in the volume description.

End of Data

The end of the volume data section is determined solely by the voxel data size and the sizes of all Data descriptors in the volume description. The descriptor for the next volume (if any) must start with a new start-of-section marker; possibly after some extraneous and non-interpreted characters.

Appendix A: Recognized Names

The following is a list of recognized names:

Recognized Names

Name	Value(s)	Interpretation
Vox1999a		Signature
Title	rest-of-line	
Copyright	rest-of-line	
//	rest-of-line	Comments
##		Start of Section Marker
VolumeCount	int	
VolumeSize	int int int	
VolumeScale	float float float	
VolumePosition	float float float	
Endian	L OR B	
Attribute	word rest-of-line	
Data	word int	
Field n	(Position int Size int Name word [Format word] [Offset float] [Scale float] [Description quoted-string])	
ModelMatrix	(float float float float float float float float float float float float float float float float)	
##\f		End of Section Marker

Appendix B: Revision History

04/18/1998 (TCZ) Initial draft and basic structure definition.

02/05/1999 (TCZ) New draft based on software teams feedback on an early version of the proposal dated 04/18/1998.

02/18/1999 (TCZ) Based on software team's review (02/17/1999), removed trailer section, made ModelMatrix 4x4, added title and copyright to volume descriptor, added start-of-descriptor marker.

3/1/1999 (AFV) Cleaned up some small problems with the text. For example, made header text start with a capital letter. Changed explicit “\n” and “newline” text to “end of line.” Changed “ASCII” to “ISO Latin-1.” Reorganized list of recognized names. Added space between “Field” and field number. Changed “name-value pair” to “descriptor.” Changed “Volume Descriptor” to “Volume Description.” Changed all ‘code’ text to Character Tag “code,” and changed it all to size 12.

6/17/1999 (AFV) Changed ISO Latin-1 back to ASCII. Added Attribute to file header and volume description. Changed text to tables for data types and field specifiers. Coined ‘field specifiers’ to refer to the name-value pairs in a Field descriptor. Clarified that comments cannot appear after a descriptor. Expressly allowed new lines inside parentheses in Field and ModelMatrix descriptors. Added Data descriptor to file header, which now requires an end-of-header marker (##\f). Added lots of MUSTs and SHOULDs to describe required, suggested, and non-required support. Did not remove field number from the Field descriptor.

7/15/1999 (AFV) Allowed for a word to be either a sequence of non-blank characters or a quoted-string. Permitted the open parenthesis of Field and ModelMatrix descriptors to be on a subsequent line, rather than requiring it to be on the first line. Changed the ModelMatrix syntax to use spaces rather than commas; the text did and does allow either blank space or a comma to separate values. Changed the meta-names n and f to int and float. Dropped the meta-name c in favor of word. (It was used in the Endian descriptor and in the Format field specifier.) Changed string to rest-of-line.

7/26/2001 (MA, AFV) Changed owner of spec to TeraRecon. Reformatted to VolumePro 1000 documentation style. Changed “vox1999a file reader” to “**Vox1999a** file reader.” Added additional standard format names: uf, sf, ui, si. Allowed other format names -- recommended all upper case to avoid conflicts with this standard. Added location of translation terms in model matrix and suggested using them to center some point of interest in the volume. Changed “hex value 0x0a” to “ASCII value 0x0A,” etc. Changed start of descriptor to be start of section, and end of header/end of descriptor to be end of section.

