VComm Signal File Specification

SimPhonics, Inc.
3226 North Falkenburg Road
Tampa, Florida 33619
Voice (877) 205-4901 X102
FAX (813) 623-5119
CAGE: 0L4C8
Email: info@simphonics.com

Table of Contents

List of Tables

1 Preface

This document defines the VComm Signal File format.

1.1 Trademarks and Copyrights

1.2 Revision History

Version	Revision	Date
1.0	Initial Release	2009-08-26
2.0	Add transmission event record	2010-07-28

For more information on this product, go to the following:
http://www.simphonics.com

1.3 Before Reading This Document

A basic understanding of networking principles is important, as well as a good understanding of the V+ Visual Programming System, VComm, and C++.

2 VComm Signal Files

The VComm Signal File (VSF) format was developed by SimPhonics to efficiently store recordings of radio transmissions. A VComm Signal File is identified with the extension “vsf” and conforms to the format specification in this document.

VSF files are created by object 2085 – VComm Recorder. Please see the VComm User Manual and the object help for more information on this object.

VSF files may be converted to standard WAV files using the VSF Converter utility which can be read by industry software compatible with the .WAV file format.

The utility can also convert VSF files to other formats.

Please see the VComm User Manual for further details on the VSF Converter utility.

3 Format Specification

The VSF file format is defined in the VSF.h header file. This header file is maintained by SimPhonics and is reproduced in section 3.2. Please refer to the header file while reading this section.

Note that absolute time stamps are stored using the Windows FILETIME structure. A FILETIME structure contains a 64 bit value representing the number of 100 nanosecond intervals since January 1, 1601 (UTC).

Note also that Intel byte order is used.

3.1 Format Description

A VSF file is a binary file that starts with a preamble and contains one or more VSF records.

3.1.1 VSF Preamble

Each VSF file starts with a preamble which contains the following fields:

- A 32 bit signature set to 0x46535600: This signature should be checked before processing a VSF file to verify the integrity of the file.
- A 32 bit version that defines the version of the file format.
- The time when the recording started.

3.1.2 VSF Records

After the preamble, the remainder of the VSF file is made up of VSF records. There are several types of VSF records but each type starts with the following fields:

- A 32 bit size that contains the total number of bytes in the record including the size field itself and any variable length data at the end of the record.
- A 32 bit unsigned integer id to specify the type of the record.
- A timestamp that defines the absolute time at which the event represented by the record occurred.

The types and descriptions of VSF records are listed in Table 1. Some records do not require any further data (i.e. they simply record the time at which the event occurred). Others, however, contain additional data which is described in the subsections below.

Table 1. VSF Record Types

ID	Record Type	Description
0	ERROR	A record type with id of 0 is an error and should be ignored.
1	AUDIO DATA UNSUPPORTED	This record represents audio data that could not be processed for some reason. For example, the encoding of the source audio data might be unsupported.
2	AUDIO DATA	This record represents a chunk of recorded audio.
3	END	This record marks the time at which the recording was stopped. This should be the last record in the VSF file.
4	PAUSE	This record marks the time at which the recording was paused.
5	RESUME	This record marks the time at which the recording was resumed.
6	TRANSMITTER STATE	This record represents the current state of a transmitter.

3.1.2.1 Audio Data Record

The Audio Data record contains time stamped recorded audio data. In addition to the initial common fields, the following fields make up the record:

- The sample rate of the recorded audio data in Hz (e.g. 8000).
- The site number, application number, entity number, and radio number of the radio that transmitted the audio data.
- A variable number of bytes that represents the recorded audio. The size of the recorded audio can be inferred from the size of the record. The audio data is stored in 16 bit stereo PCM.

3.1.2.2 Resume Record

The Resume record defines the absolute time at which recording was resumed. In addition to the initial common fields, the following fields make up the record:

- A position offset in milliseconds from the start of the recording. A value of zero or more indicates that, within the context of an exercise or simulation, the recording was resumed at a different time than when it was paused. A negative value indicates that the recording resumed from the point at which it was paused.

An example is helpful to understand the use of the position field. Consider a recorded exercise that runs for an hour and then is paused. If the exercise is rewound to the 10 minute mark and then resumed, the Resume record can store this information with a value of 600,000 in the position field (600,000 milliseconds = 10 minutes).

3.1.2.3 Transmitter State Record

The Transmitter State record contains the time stamped transmission state of a radio. In addition to the initial common fields, the following fields make up the record:

- The site number, application number, entity number, and radio number of the radio
- The current state of the radio.

The state of the radio is an enumerated value as follows:

- 0 – Radio is turned off.
- 1 – Radio is turned on but is not transmitting.
- 2 – Radio is turned on and is transmitting.
- 3 – Radio has been deleted (i.e. the radio has stopped reporting its state)

3.2 VSF.h

```
////////////////////////////////////////////////////////////////////////////
//
// VSF.h - This file defines the VComm Signal File format.
//
// The VComm Signal File Format is used to record incoming radio transmissions
// and associated events and data.
//
////////////////////////////////////////////////////////////////////////////
//
//  Copyright © 2010 SimPhonics, Inc. All rights reserved.
//  PART NUMBER:     (Same as filename)
//    ORIGINAL AUTHOR:  Richard Fedorowicz,  04/12/09 (Version 1)
//       MODS:    Richard Fedorowicz, 07/14/10, Add Transmitter State Record (Version 2)
//
////////////////////////////////////////////////////////////////////////////
#pragma once            // Include only once
#pragma pack(push, 1)       // packing is now 1

#define VSF_SIGNATURE         0x46535600
#define VSF_VERSION           2

#define VSF_MAX_DATA_SIZE     65536 // maximum amount of variable length data in a record

// VSF Preamble - Each VSF file starts with a preamble
typedef struct {
   unsigned __int32  signature;        // identifies that this is a VSF file
   unsigned __int32  version;       // VSF format version
   FILETIME       startTime;        // start time of recording
} VSF_PREAMBLE, *LP_VSF_PREAMBLE;

// VSF Records - The preamble is followed by one or more records
typedef enum { VSF_RECORD_TYPE_ERROR = 0,
            VSF_RECORD_TYPE_AUDIO_DATA_UNSUPPORTED,
            VSF_RECORD_TYPE_AUDIO_DATA,
            VSF_RECORD_TYPE_END,
            VSF_RECORD_TYPE_PAUSE,
            VSF_RECORD_TYPE_RESUME,
            VSF_RECORD_TYPE_TRANSMITTER_STATE
} VSF_RECORD_TYPE;

// VSF Record Header
typedef struct {
   unsigned __int32  size;          // size of record including data if any
   unsigned __int32  recordType;       // type of record
   FILETIME       timestamp;
} VSF_RECORD_HEADER, *LP_VSF_RECORD_HEADER;

// VSF Audio Data Record - Recorded radio transmission audio
typedef struct {
   VSF_RECORD_HEADER hdr;
   ULONG          sampleRate;       // Sample rate of audio
   USHORT            site;          // Site number of transmitting radio
   USHORT            application;      // Application number of transmitting radio
   USHORT            entity;           // Entity number of transmitting radio
   USHORT            radio;            // Radio number of transmitting radio
                                 // variable amount of data follows here
} VSF_AUDIO_DATA_RECORD, *LP_VSF_AUDIO_DATA_RECORD;

// VSF End Record - Marks the end of a recording
typedef VSF_RECORD_HEADER VSF_END_RECORD, *LP_VSF_END_RECORD;

// VSF Pause Record - Marks a pause point in a recording
typedef VSF_RECORD_HEADER VSF_PAUSE_RECORD, *LP_VSF_PAUSE_RECORD;

// VSF Resume Record - Marks a resume point in a recording
typedef struct {
   VSF_RECORD_HEADER hdr;
   LONG           position;         // The offset in milliseconds from the start of the recording
                                 // where recording resumes.  Negative values = append at end
} VSF_RESUME_RECORD, *LP_VSF_RESUME_RECORD;

// VSF Transmitter State Record - Reports the current state of the transmitter
typedef struct {
   VSF_RECORD_HEADER hdr;
   USHORT            site;          // Site number of radio
   USHORT            application;      // Application number of radio
   USHORT            entity;           // Entity number of radio
   USHORT            radio;            // Radio number of radio
   USHORT            state;            // The state of the radio transmitter
} VSF_TRANSMITTER_STATE_RECORD, *LP_VSF_TRANSMITTER_STATE_RECORD;

// Radio Transmitter States
typedef enum { VSF_TRANSMITTER_STATE_OFF = 0,
            VSF_TRANSMITTER_STATE_ON,
            VSF_TRANSMITTER_STATE_ON_AND_TRANSMITTING,
            VSF_TRANSMITTER_STATE_DELETED
} VSF_TRANSMITTER_STATE;

#pragma pack(pop)
```