New Applications of the Sound Description Interchange Format

Matthew Wright, Amar Chaudhary, Adrian Freed, David Wessel, {matt,amar,adrian,wessel}@cnmat.berkeley.edu
CNMAT, 1750 Arch St., Berkeley, CA 94709 USA

Xavier Rodet (rod@ircam.fr), Dominique Virolle (virolle@ircam.fr), Rolf Woehrmann*
IRCAM, 1 Place Igor Stravinsky, 75004 Paris, France
* Currently at Steinberg, Postfach 26 1833, D-20508 Hamburg, Germany (Rolf.Woehrmann@steinberg.de)

Xavier Serra (xserra@iua.upf.es), Audiovisual Institute, Pompeu Fabra University

The Sound Description Interchange Format (SDIF) is a recently adopted standard that can store a variety of sound representations including spectral, time domain, and higher-level models. SDIF consists of a specified data format and a set of standard sound descriptions and their official representation. SDIF is flexible in that new sound descriptions can be represented, and new kinds of data can be added to existing sound descriptions, facilitating innovation and research. This paper describes the goals and design of SDIF and its standard frame types, followed by a review of recent SDIF work at CNMAT, IRCAM, and IUA.

1. Goals of SDIF

Many groups around the world are doing interesting work with descriptions of sound other than the ubiquitous time-domain sample representation. Unfortunately, we each have our own incompatible file formats, making it difficult to share results, tools, data, etc. The main goal of SDIF is to promote interchange by providing a common format for a variety of sound descriptions.

Many of SDIF's features support this goal. SDIF is multi-platform and uses standard data types such as ASCII and big-endian IEEE 754 floats and two's complement integers. The SDIF format is reasonably space efficient and conceptually straightforward. The SDIF specification is open and publicly available (http://www.cnmat.berkeley.edu/SDIF and http://www.ircam.fr/equipes/analyse-synthese/sdif), and defines semantics for interpreting SDIF data. Another goal is a forthcoming C and C++ library, to be available at no charge.

SDIF's design strikes a balance between an overly strong standard, which would restrict innovative and creative uses, and an overly weak standard, in which it is so easy to modify the format that each institution has its own incompatible version of SDIF, even when using the same sound representation. SDIF has standardized formats to support common extant sound descriptions, described below. SDIF allows custom versions of these representations that include all the standard data in the standard places, plus extra fields, for example, the perceptual salience of each partial in a resonance model or the "noisiness" of each spectral peak. Entirely new representations can also be added. In all cases a program that manipulates standard SDIF data may safely ignore unrecognized custom information.

SDIF was designed to support file storage and Internet streaming of aggregates of various sound descriptions. (In this paper, the term "SDIF file" will mean either a normal file or a stream of SDIF data.)

We hope that SDIF will encourage and facilitate the development of new tools for manipulating sound in spectral and other domains, promote the use of interesting sound descriptions in general, and facilitate sharing of work within the community.

2. The SDIF Format

The SDIF format is a sequence of *frames*, modeled after chunks in the IFF/AIFF/RIFF formats. To help guarantee that all 8-byte data are aligned at the beginning of an 8-byte word (and similarly for 4-byte data), every frame's length is a multiple of eight bytes. A frame consists of the following:

- FrameTypeID, a unique code indicating what kind of frame this is
- FrameSize, the size in bytes of the frame, not counting the FrameTypeID or FrameSize
- The actual data
- ZeroPadding, enough zero bytes to make the total size of the frame be a multiple of eight bytes

An SDIF file must begin with a small opening frame that identifies the file as SDIF. The FrameTypeID for this frame is "SDIF," so all SDIF files start with those four bytes. A few optional frame types may appear immediately after the opening frame; these document or explain how to interpret the main part of the SDIF data. Most SDIF data, however, is self-explanatory, so these frame types are rarely needed.

The body of the file is a contiguous sequence of time-tagged frames, sorted in ascending temporal order. The time tag is an eight byte floating point number that indicates the time to which the given frame applies. The units are seconds and the time tag can be negative. Most frame types are applicable to a single instant of time, for example, the center point of a STFT window. A few frame types, for example, blocks of time-domain envelope breakpoints or samples, represent data that span a given interval of time; in these cases the frame's time tag indicates the beginning of the interval. Since every frame of data has its own time tag, SDIF data is in general sampled at arbitrary time points.

An SDIF *stream* is a series of frames with different time tags that represent the same sonic object over time. In this view, the sequence of all frames can be considered a set of interleaved streams. Each frame in the body of an SDIF file has a StreamID; frames with the same StreamID belong to the same stream.

A single SDIF file may contain multiple kinds of frames; in this sense SDIF can be an aggregate or archive format. A library of standard frame types defines formats for storing common sound representations that are part of the SDIF standard.

Data in almost all frame types are stored in one or more 2D *matrices*. Matrix columns correspond to parameters like frequency or amplitude; each row represents an object like a filter, sinusoid, or noise band. A matrix consists of the following:

- A matrix type ID, similar to the FrameTypeID
- A count of the size, in bytes, of the matrix
- A code for the type of the data in the matrix (for example, 4 or 8 byte integer or floating point number)
- The numbers of rows and columns in the matrix
- The data themselves
- Optional byte padding to make the overall matrix size a multiple of 8 bytes

Most frame types will consist of a single matrix. Some frame types will consist of a main matrix of data plus a few extra fields in a secondary 1D matrix, e.g., STFT frames must include the frequency sampling interval and the frame size as well as the actual STFT bin values.

Extra columns can be added to standard matrix types to encode custom information like the “width” of a spectral line-broadened sinusoid [Risset 82, Fitz 95, Freed 98] or the perceptual salience of a particular filter.

3. Standard SDIF Frame Types

The part of SDIF that actually defines certain sound descriptions is the library of standard frame types. Each of these frame types has a unique FrameTypeID beginning with ASCII “1,” indicating the first version of the specification. Each standard frame type also has one or more standard matrix types that must appear in frames of that type; the standard gives the semantics of the rows and columns of those matrices.

The frame type for fundamental frequency estimates (“1FQ0”) contains a single matrix whose rows are candidate fundamentals suggested by the estimator and whose columns are the frequency (in Hertz) and a confidence factor.

1STF frames represent a block from a Discrete Short-Time Fourier Transform. The time tag in a 1STF frame is the time of the *center* of the window, not the beginning. These frames consist of two matrices. One matrix records overall information about the transform: the frequency sampling interval and the length in seconds of the time-domain information that was the input to the transform. The main data matrix has a row for each frequency bin output by the transform and columns for real and imaginary parts.

Picked spectral peaks frames (“1PIC”) represent local maxima in a spectrum at a given time. They contain a single matrix with rows for each peak and columns for amplitude, frequency, phase, and a confidence factor that might be used to indicate how much of the energy around this peak was from a sinusoid or how well the energy around this peak matches a sinusoid.

Sinusoidal track frames (“1TRC”) represent sinusoids that maintain their continuity over time as their frequencies, amplitudes, and phases evolve. They contain a single matrix with rows for each sinusoid and columns for amplitude, frequency, phase, and an index that identifies each track and allows it to be matched with 1TRC data in other frames. Harmonic sinusoidal track frames (“1HRM”) differ from 1TRC frames only in that the partials are understood to lie on or close to a harmonic series. Thus, the index column of the 1HRM matrix represents harmonic number rather than an arbitrary index.

Resonance [Potard 86] frames (“1RES”) contain a single matrix with rows for each resonance and columns for amplitude, frequency, decay rate, and initial phase of a set of exponentially decaying sinusoids.

Noise frames (“1NOI”) can contain several matrix types. The “1DIS” matrix gives the distribution(s) of the random signal, for example, uniform, Gaussian, etc. Another matrix represents parameters for a synthesis model of noise filtered into frequency bands controlled by amplitude envelopes. It has rows for each noise band and columns for amplitude and for the upper and lower edges of the frequency range of the band.

4. Uses of SDIF at CNMAT, IRCAM, and IUA

The SDIF library

At IRCAM and CNMAT, the analysis/synthesis teams have decided to use SDIF for all new software developments and gradually to upgrade existing software to use SDIF. For this purpose, we have written an SDIF library for reading and writing SDIF files. This library is currently used on multiple platforms, including SGI IRIX, DEC Alpha OSF, Apple MacOS, Windows, and Linux, and is available to the public at no charge.

The SDIF library contains groups of functions for reading and writing name/value tables, stream info blocks, frames, and matrices. These functions hide the details of the SDIF byte format and automatically handle issues such as frame headers and eight byte alignment. The library also includes abstract numeric types of the required sizes for SDIF data that work on each architecture.

The SDIF library incorporates a flexible strategy for dealing with new frame and matrix types and new columns in standard matrix types. A special extension file named “SdifTypes.STYP” defines these customized or new frame and matrix types in a simple ASCII text format. This allows the library to handle new types without recompilation and facilitates experimentation with new types. Here is an example of an “SdifTypes.STYP” file:

```
1TYP {
  1MTD  1FOF    {Freq, Amp, DecayRate, Phase, ExcitationTime, SquelchStartTime, SquelchRate}
  1MTD  1FQ0    {Freq, Mode, Hit}
  1FTD  1FOB    {1FQ0 PitchModeHit; 1FOF Formants;}
}
```

The “1MTD” lines define new matrix types by giving the MatrixTypeID and a list of the names of the columns in the matrix. The code “1FTD” defines a new frame type, which consists of the FrameTypeID followed by a list of the matrices that appear in the frame.

SDIF/Text Conversion Utilities

We have created an invertible SDIF to text transformation that allows users to view the contents of an SDIF file and to use a text editor to modify SDIF data or create new data by hand [Virolle 98]. Two simple applications, “sdifToText” and “textToSdif,” provide conversion between an SDIF file and this representation and facilitate debugging of SDIF applications.

Here is an excerpt of the textual representation of an SDIF file. Note the FrameTypeIDs in the leftmost column, and the use of indentation to show the structure of matrices within frames.

```
1FOB    2       1       0.
  1FQ0  32      1       1
        110.
  1FOF  32      2       7
        650.    1       80.     0.002   0.05    0.004   0.
        1080.   0.5012  90.     0.002   0.05    0.004   0.
```

CHANT

IRCAM's new CHANT synthesizer (http://www.ircam.fr/produits-real/logiciels/chant.html) uses SDIF via a new library written by D. Virolle (http://www.ircam.fr/equipes/analyse-synthese/libchant). This synthesizer reads its configuration from an SDIF file. Here is an example that configures Chant and defines the given SDIF stream to address a particular formant waveform (FOF) in a given FOF bank:

```
          0 Chant:Patch0/1/FOB/1/5/0./2.5;
```

This library was ported to the Macintosh by Adrien Lefevre and is included as a plug-in component in the Diphone software [Rodet 97], which uses SDIF files to store and read Chant filters and (FOF) data.

Spectral Envelopes

A project at IRCAM uses SDIF to store the time-varying spectral envelope of a sound [Schwartz 98]. There is an individual frame type for each of the different parameterizations of spectral envelopes obtained by various estimation methods used in the project: cepstrum and discrete cepstrum coefficients, autoregressive coefficients, reflection coefficients, formant, “fuzzy formants,” discretized spectral envelopes, etc. The spectral envelope of the sinusoidal component and of the residual noise component are stored in a single file using SDIF’s stream facility.

D. Schwarz has written programs which apply SDIF-encoded spectral envelopes to several sound representations, including time-domain sample files, spectral peaks, sinusoidal tracks, resonances, and noise bands.

SMS

The research group at the Audiovisual Institute (IUA) of the Pompeu Fabra University uses SDIF to communicate among analysis programs based on the SMS technique [Serra 97], synthesizers, and graphical editors. A complete description of the frame types developed at the IUA is available (http://www.iua.upf.es/~sms), including:

- Generic: Container frame for one or several tracks.
- Generic Track: Container frame with some global information for the track and pointing to other frames.
- Region: Frame that describes a segment of a track with common characteristics.
- Envelope: Frame used to store High Level Attributes of a Generic Track, a Region, or another frame.

OpenSound Edit

OpenSoundEdit (OSE) [Chaudhary 98] is a tool for visualizing and editing SDIF data, using a binding of the SDIF library to Tcl [Welch 97]. OSE currently includes visualization methods for resonance and sinusoidal track models.

CAST

CNMAT's real-time additive synthesizer, “softcast” (http://www.cnmat.berkeley.edu/CAST), reads tracks of partials, pitch estimates, noise data [Freed 98] and resonance data from SDIF files. It synthesizes from these data types offering extensive control over the progression through the time axis of the SDIF file with respect to real time and numerous real-time timbral modifications including inharmonicity, filtering, and timbral interpolation.

5. Conclusion

It would appear from the applications developed thus far that SDIF has gained ground as a standard. SDIF has demonstrated its utility sufficiently that it will be used for future projects, and we hope to see SDIF adopted by other groups as well.

6. References

[Chaudhary 98] A. Chaudhary, A. Freed, L. Rowe. “OpenSoundEdit: An Interactive Visualization and Editing Framework for Timbral Resources.” Proc. ICMC98, Ann Arbor, MI, 1998.

[Fitz 95] Fitz, K., and L. Haken, “Bandwidth Enhanced Sinusoidal Modeling in Lemur,” pp. 154-157, Proc. ICMC95, Banff, Canada.

[Freed 98] Freed, A. “Real-Time Inverse Transform Additive Synthesis for Additive and Pitch Synchronous Noise and Sound Spatialization.” AES 104th Convention, San Francisco, 1998.

[Potard 86] Potard Y., Baisnée P-F., Barrière J-B., (1986), “Experimenting with Models of Resonance Produced by a New Technique for the Analysis of Impulsive Sounds,” pp.269-274, Proc. ICMC86, La Haye.

[Risset 82] Risset, J. C. and D. Wessel. “Exploration of Timbre by Analysis and Synthesis,” in D. Deutsch, editor, *The Psychology of Music*, Academic Press, 1982.

[Rodet 97] Rodet, X., and A. Lefevre, “The Diphone program: New features, new synthesis methods, and experience of musical use,” pp. 418-421, Proc. ICMC97, Thessaloniki, Hellas.

[Schwartz 98] Schwarz, D., “Spectral Envelopes in Sound Analysis and Synthesis,” Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 1622, Stuttgart, Germany, June 1998.

[Serra 97] Serra, Xavier. “Musical Sound Modeling with Sinusoids plus Noise,” in G. D. Poli, A. Picialli, S. T. Pope, and C. Roads, editors, *Musical Signal Processing*. Swets & Zeitlinger Publishers, 1997.

[Virolle 98] Virolle, D. *Chant and SDIF libraries and a Chant synthesizer.* Rapport de stage, Ircam , Janvier 1998.

[Welch 97] B. B. Welch. *Practical Programming in Tcl & Tk*, 2nd Edition. Prentice Hall, New Jersey, 1997.