1 Introduction

The Sound Effects Player is a component of ShowControl, a project to automate the production of live stage plays. Sound Effects Player makes sounds at appropriate times in the performance.

There are two aspects to the use of the Sound Effects Player. The sound designer must create the sounds, and the sound effects operator must play them during the performance. The Sound Effects Player tries to make the latter job as simple as possible, at the cost of making the sound designer's job complex. This document is intended to support the sound designer by describing how to create sounds for the Sound Effects Player.

1.1 Installation

The Sound Effects Player is installed in the traditional GNU/Linux fashion: acquire the kit, place it in a subdirectory, install the prerequisite packages, then

```
./configure --prefix=/usr
make
sudo make install
```

The prerequisite packages for Fedora are gcc, intltool, gstreamer1-devel, gstreamer-plugins-base-devel, gtk3-devel, gtk-doc and libtime. Other distributions of GNU/Linux may have other names for these packages. See the .spec file for details.

1.2 Sample Application

There is a MAN page for Sound Effects Player which provides more detail, but briefly you must provide the Sound Effects Player with a file which describes how the stage play is to be automated. The distribution kit contains a sample subdirectory which plays a single sound with no operator interaction, then exits. You can verify that you have installed the Sound Effects Player successfully by running the sample application:

```
cd sample
bash run_sample.sh
```

You should hear a 440 Hz sine wave that lasts 12 seconds and fades up and down during that time. We will examine each of the files in this subdirectory to see how they work together to make this sound. Here is the list of files and a brief description of their meaning:

File name	Description
440Hz.wav	3 seconds of a 440Hz sine wave, 8000 samples per second, 8 bits per sample
Makefile	Used by the build software

File name	Description
Makefile.am	Used by the build software
Makefile.in	Used by the build software
run_sample.sh	Program to run the sample application.
Sample_cues.xml	Dummy file which would contain the cues if this were a complete ShowControl project
Sample_equipment.xml	Pointer to the sounds and sound_sequence files; in a complete ShowControl project this file would also contain information about other equipment
Sample_project.xml	The top-level ShowControl project file
Sample_script.xml	Dummy file which would contain the script if this were a complete ShowControl project
Sample_sound_sequence.xml	The procedure for presenting sounds to the sound effects operator
Sample_sounds.xml	The sounds to be presented

The file run_sample.sh invokes sound_effects_player to play the sound specified by Sample_project.xml. Sample_project.xml consists mostly of comments explaining what each part does. You can read them to get more background on the ShowControl project, but here we are concerned just about the sound effects, so we only need to note that the equipment is described in the file Sample_equipment.xml. Sample_equipment.xml, in its sound effects section, refers to two other files: Sample_sounds.xml and Sample_sound_sequence.xml. These are the two files we will use to illustrate how to create sound effects for your live stage play. You should, of course, create files of your own for your play, but you can use the sample files to get started.

1.2.1 Sample_sounds.xml

The sounds XML file, in the sample application called Sample_sounds.xml, contains the information about each sound. Here is the Sample_sounds.xml file. The letters at the right margin are callouts referenced in the paragraphs below.

```
<?xml version="1.0" encoding="utf-8"?>
<show_control>
```

```
<!-- The sample production uses this sound. -->
  <sounds>
    <version>1.0</version>

    <!-- a sine wave at 440Hz, which is A in octave 6. -->
    <sound>
      <!-- The internal sequencer references sounds by name. -->
      <name>440Hz</name>
      <!-- the WAV file contains the waveform for the sound -->
      <wav_file_name>440Hz.wav</wav_file_name>
      <!-- volume ramps up from 0 to attack level in attack duration time -->
      <attack_duration_time>2.000000000</attack_duration_time>
      <attack_level>1.00</attack_level>
      <!-- after reaching attack level, volume ramps down to sustain level
         in decay time -->
      <decay_duration_time>1.000000000</decay_duration_time>
      <sustain_level>0.5</sustain_level>
      <!-- release starts on external signal or when reaching the
         release start time -->
      <release_start_time>10.000000000</release_start_time>
      <!-- upon release, volume ramps down to 0 in release duration time -->
      <release_duration_time>2.000000000</release_duration_time>
      <!-- sound loops between loop times; from=0 means no looping -->
      <loop_from_time>1.000000000</loop_from_time>
      <loop_to_time>0.000000000</loop_to_time>
      <!-- automatically stop looping after loop_limit loops -->
      <loop_limit>0</loop_limit>
      <!-- start at start_time offset in the wave file -->
      <start_time>0.000000000</start_time>
      <!-- attenuate the full-volume sound in the WAV file by
         designer_volume_level -->
      <designer_volume_level>1.00</designer_volume_level>
      <!-- pan mono sound or balance stereo sound by designer_pan -->
      <designer_pan>0.00</designer_pan>
      <!-- if no internal sequencer, MIDI program and note numbers to activate
         from an external sequencer  -->
      <MIDI_program_number>0</MIDI_program_number>
      <MIDI_note_number>69</MIDI_note_number>
      <!-- if no internal sequencer, OSC name to activate from an external
         sequencer  -->
      <OSC_name></OSC_name>
      <!-- if no internal sequencer, function key to activate by sound
         effects operator -->
      <function_key></function_key>
    </sound>

    <routing></routing>
  </sounds>
</show_control>
```

A
B
C
D
E
F
G
H
I
J
K

The sample application contains only one sound, which it calls 440Hz at A. Each sound refers to a WAV file which gets played when required. See B. In the simplest case the WAV file is just played once at full volume, but the Sound Effects Player can also repeat a portion of the sound, and can apply an amplitude envelope to it. In the Sample_sounds.xml file you see that the 440Hz sine wave spends

two seconds (C), reaching full volume (D), one second (E) decaying to half volume (F), then, if it hasn't been stopped by the sound effects operator, it stops itself after playing for 10 seconds (G). When it is stopped its volume decreases to 0 in two seconds (H). To see how the operator can stop a sound early, run run_sample.sh again but this time, instead of letting it finish by itself, click on the Stop button in cluster 0 and observe that the total run time is now less than 12 seconds.

The sample sound also saves memory and disk space by looping. If you examine the file 440Hz.WAV you will see that it is only three seconds long. The information in the <sound></sound> block specifies that the first second (I), (J) of the WAV file is to be repeated indefinitely until the sound is released (K). The sound is released when it has been playing for 10 seconds (G) or when the sound effects operator presses the Stop button.

If you examine the 440Hz.WAV file you will see that it has only 8 bits per sample, and only 8,000 samples per second. That is enough for a simple sine wave. The sound effects player uses the Gstreamer structure to accept sample rates from 6,000 to 96,000 samples per second, using formats U8 (8 bits per sample), S16LE (16 bits per sample), S32LE (32 bits per sample), F32LE (32 bits per sample floating-point) and F64LE (64 bits per sample floating-point).

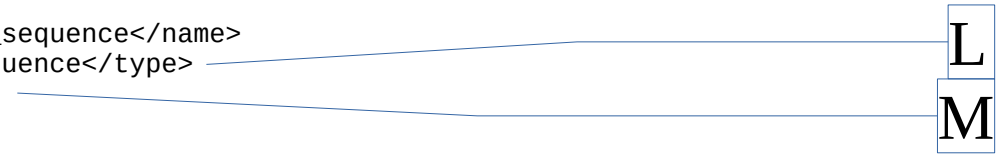
I use audacity to prepare WAV files for the Sound Effects Player. It provides all the flexibility I need, but is simple to use. Other tools are also available: I have used espeak and sox to prepare spoken words, for example.

1.2.2 Sample_sound_sequence.xml

The sound sequence XML file, in the sample application called Sample_sound_sequence.xml, presents the sounds to the sound effects operator. Some parts of a live stage play can be handled by sequential processing: simply making the next sound when the time comes to make it. Some parts, however, are best handled by presenting the sound effects operator with a selection of sounds which he triggers in response to activities on the stage. The sound effects player combines these methods in a very flexible way.

The Sample_sound_sequence.xml file simply plays the 440Hz sound without waiting for the operator. Here it is:

```
<?xml version="1.0" encoding="utf-8"?>
<show_control>

  <!-- The sequence items for the sample production.  -->

  <sound_sequence>
    <version>1.0</version>

    <sequence_item>
      <name>start_of_sequence</name>
      <type>start_sequence</type>
```



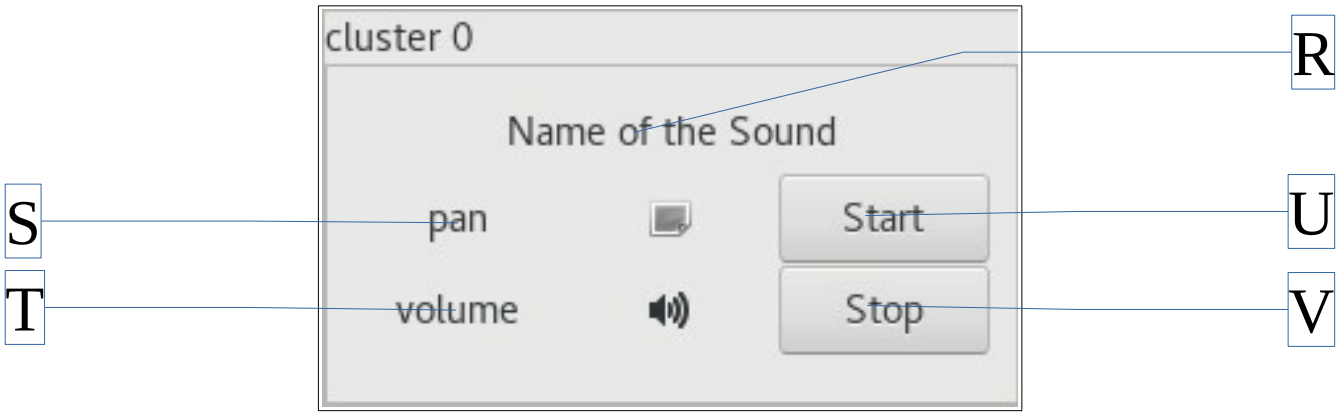
```
      <next>A6</next>
    </sequence_item>

    <sequence_item>
      <name>A6</name>
      <type>start_sound</type>
      <sound_name>440Hz</sound_name>
      <MIDI_note_number>69</MIDI_note_number>
      <cluster_number>0</cluster_number>
      <tag>A6</tag>
      <text_to_display>A in octave 6</text_to_display>
      <importance>1</importance>
    </sequence_item>

  </sound_sequence>
</show_control>
```

N O P Q

A sound sequence file, in this sample program called Sample_sound_sequence.xml, contains sequence items. Each item performs a function when executed by the Sound Effects Player. Every sequence item has a name, a type and whatever other information that it needs to carry out its function. The sample sequence consists of just two sequence items: a start_sequence and a start_sound.
Start_sequence (L) is where the Sound Effects Player starts. It then moves to the sequence item named in the <next></next> block (M), in this case A6. The sequence item named A6 (N) starts playing the sound named 440Hz (O), which we saw in Sample_sounds.xml (A). It specifies that the sound effects operator is to have control of the sound using cluster number 0 (P), and the text to display to the sound effects operator is “A in octave 6” (Q). This sequence item does not specify what the Sound Effects Player is to do when the sound finishes playing, so the Sound Effects Player exits.

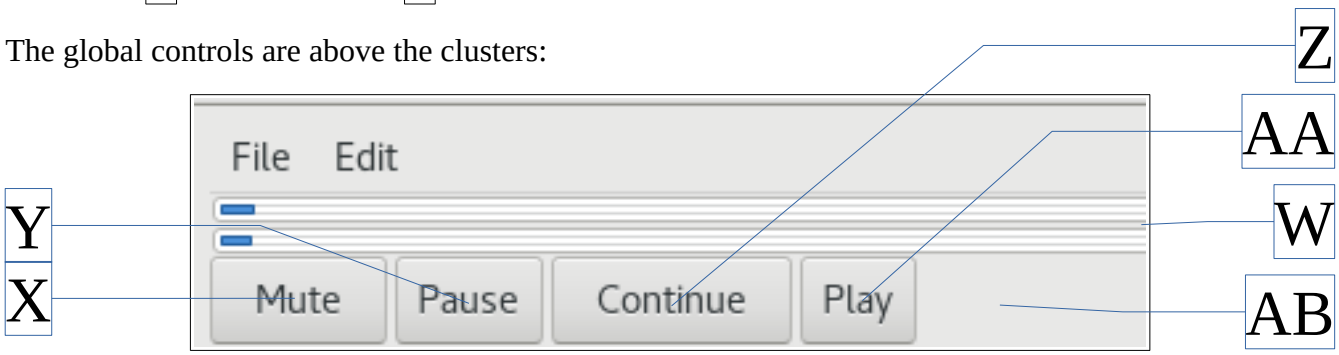
2 Clusters

The sample application lets the sound effects operator control its sound using cluster 0. The Sound Effects Player displays 16 clusters, numbered 0 through 15. A cluster looks like this:



The text area across the top of the cluster (R) is controlled by the sequence item (Q). The four controls beneath are for panning the sound through the stereo field (S), adjusting its volume (T), starting it (U) and stopping it (V).

The global controls are above the clusters:



Above the buttons are horizontal bars that show how loud each channel is sounding (W). From left to right, the buttons silence the output (X), pause the sound (Y), continue from a pause (Z) and continue from an Operator Wait (AA). To the right of the Play button is a text area (AB) which is filled in by the Operator Wait. We will see the Operator Wait sequence item in the next section.

3 Example_01

To illustrate more of the capabilities of the Sound Effects Player we need a more complex example. Navigate to subdirectory example_01 and you will find file run_example_01.sh. Running this file executes the sound effects player, giving it Example_01_project.xml as its parameter. Type "./run_example_01.sh" to see what this project does.

Notice that it plays the 440Hz tone, then waits for you to press a key. The text to the right of the Play button (AB) is "Press a Start key to hear a number, Play to exit". Each of the clusters has some text in its display area (R) which invites you to press the Start button (U) and describes what you will hear when you do. Press some of the Start buttons to hear a synthesized voice saying numbers between one and sixteen. Notice that the text area over the cluster changes while the voice is speaking. When you have had enough, press the Play button in the global controls (AA) to cause the Sound Effects Player to exit.

The XML files in the example_01 subdirectory are structured very much like the files in the sample subdirectory, so we only need to be concerned with files example_01_sounds.xml and example_01_sound_sequences.xml. Example_01_sounds.xml contains, in addition to the 440Hz sound, sixteen sounds that look like this:

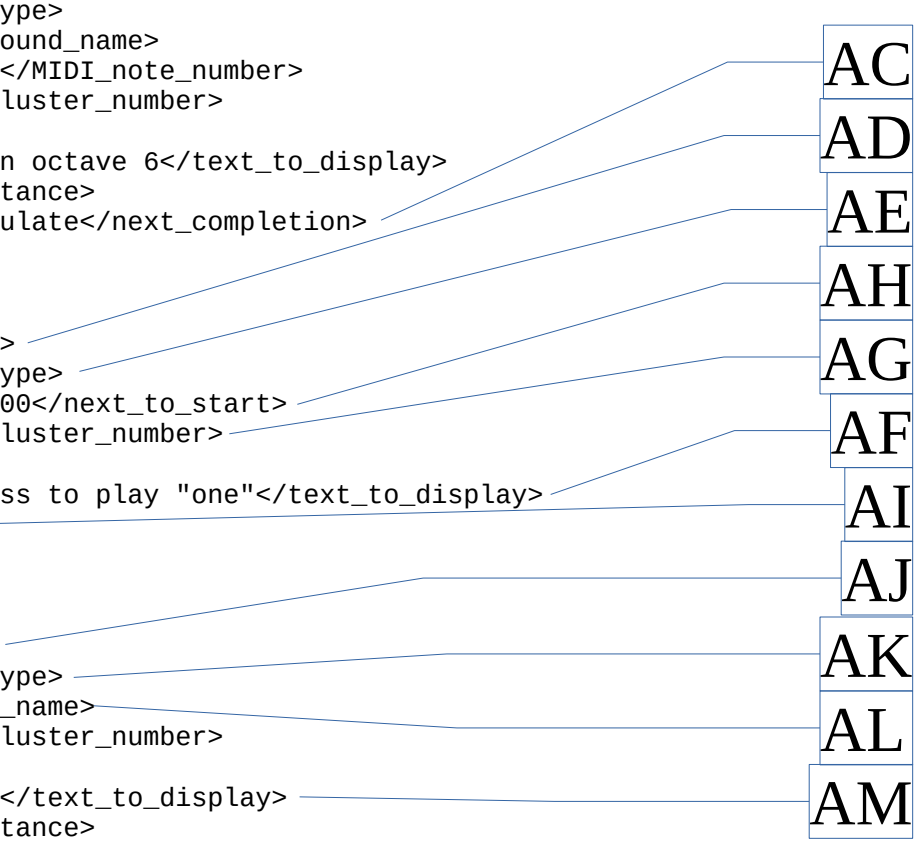
```
<sound>
      <!-- The internal sequencer references sounds by name. -->
      <name>1</name>
```

```
    <!-- the WAV file contains the waveform for the sound -->
    <wav_file_name>01.wav</wav_file_name>
  </sound>
```

Each block associates a sound name with a WAV file name, and lets all of the other sound parameters default. The default values play the sound once, at full volume.

Example_01_sound_sequence.xml starts like this:

```
<?xml version="1.0" encoding="utf-8"?>
<show_control>

  <!-- The sequence items for the example_01 production.  -->

  <sound_sequence>
    <version>1.0</version>

    <sequence_item>
      <name>start_of_sequence</name>
      <type>start_sequence</type>
      <next>A6</next>
    </sequence_item>

    <sequence_item>
      <name>A6</name>
      <type>start_sound</type>
      <sound_name>440Hz</sound_name>
      <MIDI_note_number>69</MIDI_note_number>
      <cluster_number>0</cluster_number>
      <tag>A6</tag>
      <text_to_display>A in octave 6</text_to_display>
      <importance>1</importance>
      <next_completion>populate</next_completion>
    </sequence_item>

    <sequence_item>
      <name>populate</name>
      <type>offer_sound</type>
      <next_to_start>play_00</next_to_start>
      <cluster_number>0</cluster_number>
      <tag>1</tag>
      <text_to_display>press to play "one"</text_to_display>
      <next>pop_01</next>
    </sequence_item>

    <sequence_item>
      <name>play_00</name>
      <type>start_sound</type>
      <sound_name>1</sound_name>
      <cluster_number>0</cluster_number>
      <tag>1</tag>
      <text_to_display>one</text_to_display>
      <importance>1</importance>
    </sequence_item>
```



The beginning is just like Sample_sound_sequence.xml (L through Q), but notice the new field at AC. The <next_completion>populate</next_completion> field causes the Sound Effects Player to execute the populate sequence item when the sound being played has completed.

The sequence item named populate (AD) is an offer_sound (AE). It puts some text (AF) on the text area (R) of cluster 0 (AG) and tells the Sound Effects Editor to execute sequence item play_00 (AH) when the sound effects operator pushes the Start button on cluster 0 (U). It then immediately goes on to execute sequence item pop_01 (AI) which does the same thing for cluster 1, and so on through cluster 15.

Pushing the Start button on cluster 0 causes the Sound Effects Player to execute sequence item play_00 (AJ). This is a start_sound sequence item (AK) which plays sound 1 (AL) and changes the text on cluster 0's text area (AM). Reading through the remainder of Example_01_sound_sequence.xml you will see that the same happens for clusters 1 through 15.

Further down in file Example_01_sound_sequence.xml we see this:

```
    <sequence_item>
      <name>pop_15</name>
      <type>offer_sound</type>
      <next_to_start>play_15</next_to_start>
      <cluster_number>15</cluster_number>
      <tag>16</tag>
      <text_to_display>press to play "sixteen"</text_to_display>
      <next>do_wait</next>
    </sequence_item>

    <sequence_item>
      <name>play_15</name>
      <type>start_sound</type>
      <sound_name>16</sound_name>
      <cluster_number>15</cluster_number>
      <tag>16</tag>
      <text_to_display>sixteen</text_to_display>
      <importance>1</importance>
    </sequence_item>

    <sequence_item>
      <name>do_wait</name>
      <type>operator_wait</type>
      <text_to_display>Press a Start key to hear a number, Play to
exit</text_to_display>
      <next_play>clean_01</next_play>
    </sequence_item>

    <sequence_item>
      <name>clean_01</name>
      <type>cease_offering_sound</type>
      <tag>1</tag>
```

AU AN AO AP AQ AR AS

```
      <next>clean_02</next>
    </sequence_item>
```

AT

The fifteeth offer_sound branches to do_wait (AN). Do_wait (AO) is an Operator_wait sequence item which displays “Press a Start key to hear a number, Play to exit” (AP). When the sound effects operator presses the Play button, the Sound Effects Editor executes the clean_01 sequence item (AQ). The clean_01 sequence item (AR) is a cease_offering_sound which specifies that the sound offered with a tag of 1 (AS) should be canceled. It then branches to clean_02 (AT) which does the same for tag 2, and so on to tag 16.

Each of the offer_sound sequence items contains a tag (AU), which is used by cease_offering_sound to terminate the offering.

At the bottom of file Example_01_sound_sequence.xml we find this:

```
    <sequence_item>
      <name>clean_16</name>
      <type>cease_offering_sound</type>
      <tag>16</tag>
      <next>exiting</next>
    </sequence_item>

    <sequence_item>
      <name>exiting</name>
      <type>wait</type>
      <time_to_wait>2.0</time_to_wait>
      <text_to_display>exiting...</text_to_display>
    </sequence_item>

  </sound_sequence>
</show_control>
```

AV

AW

AX

AY

AZ

The last of the cease_offering_sound sequence items branches to sequence item exiting (AV). The exiting sequence item (AW) is a wait (AX) which delays for 2 seconds (AY) while it displays “exiting...” to the sound effects operator (AZ). It has no <next></next> block, so when it is complete there are no sequence items in execution, and no sounds being offered on clusters. When that happens, the Sound Effects Editor exits.

4 Example_02

Example 2 is in subdirectory example_02. It has two sounds:

- car, the sound of a car engine starting, idling, then shutting down
- circular_saw, the sound of a circular saw starting, cutting, then stopping

The file Example_02_sound_sequence.xml simply offers the two sounds. The interesting file is Example_02_sounds.xml, which looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<show_control>
<!-- The example_02 production uses these sounds. -->
  <sounds>
    <version>1.0</version>

    <sound>
      <name>circular_saw</name>
      <wav_file_name>circular_saw.wav</wav_file_name>
      <!-- The saw runs until the sound effects operator
         presses the Stop key.  -->
      <loop_from_time>2.5</loop_from_time>
      <loop_to_time>2.45</loop_to_time>
      <loop_limit>0</loop_limit>
      <release_duration_time>∞</release_duration_time>
    </sound>

    <sound>
      <name>car</name>
      <wav_file_name>car.wav</wav_file_name>
      <!-- The car runs until the sound effects operator
         presses the Stop key.  -->
      <loop_from_time>11.01925</loop_from_time>
      <loop_to_time>10.00227</loop_to_time>
      <loop_limit>0</loop_limit>
      <release_duration_time>∞</release_duration_time>
    </sound>

    <routing></routing>
  </sounds>
</show_control>
```

Cars and saws are complex sounds: they don't just go silent when they stop, but have a shutdown sound. For a live stage play we want to make the sound until the script calls upon it to stop, which will be a variable length of time depending upon the actor. Thus we play the start of the sound, repeat the middle of the sound until time to stop, and finally play the end of the sound. This is done by setting the <loop_to_time></loop_to_time> and the <loop_from_time></loop_from_time> fields to the beginning and ending of the middle part of the sound. Also, we set <loop_limit></loop_limit> to 0, meaning loop until the Stop button is pushed, and <release_duration_time></release_duration_time> to infinity (∞) so that the envelope does not ramp the volume down after the stop button has been activated.

The tricky part of such a sound is finding good loop points. To avoid a click at the transition, choose points where the waveform crosses 0 and is headed in the same direction, either increasing or decreasing. I use Audacity for this, zooming in until I can see individual samples.

Another problem is the repetition. You don't want a 1-second sound to be repeated many times unless it is a very simple sound, like a hum. The audience will quickly become annoyed. To mask the

repetition you can use a second sound which repeats at a different interval. Try playing the car and the saw together, to see how the repetition is less noticeable with both running.

5 Example_03

Example 3 is in subdirectory example_03. It illustrates how to divide sound effects into two groups: those which are used routinely in the theater, and those which are constructed especially for this production. The files in this subdirectory are as follows:

File name	Description
Automobile.wav	The sound of an automobile starting up, running, and pulling to a stop, 16 bits per sample, 44,100 samples per second.
background_music.wav	The background music for the play, stereo, 8 bits per sample, 6000 samples per second.
Example_03_cues.xml	The special cues for this production
Example_03_equipment.xml	The special equipment for this production: references Example_03_sounds.xml and Example_03_sound_sequence.xml
Example_03_project.xml	The project file for this production: references Theater_equipment.xml, Example_03_equipment.xml
Example_03_script.xml	A dummy script file for this production
Example_03_sound_sequence.xml	The special sound sequences for this production—see below
Example_03_sounds.xml	The special sounds for this production—see below
full_ring.wav	The sound of a telephone ringing once, lasts 6 seconds
Makefile	Used by the build system
Makefile.am	Used by the build system
Makefile.in	Used by the build system
run_example_03.xml	Program to run Sound Effects Player for this production
Theater_cues.xml	The regular cues for this theater
Theater_equipment.xml	The regular equipment used in this theater: references Theater_sounds.xml and Theater_sound_sequence.xml
Theater_sound_sequence.xml	The regular sound sequences for this theater—see below

File name	Description
Theater_sounds.xml	The regular sounds for this theater—see below

When the Sound Effects Player runs, it reads Example_03_project.xml, and from it loads the sounds and sound sequences in Theater_sounds.xml, Example_03_sounds.xml, Theater_sound_ssequence.xml and Example_03_sound_sequence.xml.

5.1 Theater_sounds.xml

The file Theater_sounds.xml looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<show_control>
<!-- these sounds are common to many shows in the theater. -->
  <sounds>
    <version>1.0</version>

    <!-- The ring and ringout sounds are used for a telephone. -->
    <sound>
      <name>ring</name>
      <wav_file_name>full_ring.wav</wav_file_name>
      <attack_duration_time>0.000000000</attack_duration_time>
      <attack_level>1.00</attack_level>
      <decay_duration_time>0.000000000</decay_duration_time>
      <sustain_level>1.00</sustain_level>
      <release_start_time>2.995000000</release_start_time>
      <release_duration_time>0.010000000</release_duration_time>
      <loop_from_time>0.000000000</loop_from_time>
      <loop_to_time>0.000000000</loop_to_time>
      <loop_limit>0</loop_limit>
      <start_time>0.000000000</start_time>
      <designer_volume_level>1.00</designer_volume_level>
      <designer_pan>0.00</designer_pan>
      <MIDI_program_number></MIDI_program_number>
      <MIDI_note_number></MIDI_note_number>
      <function_key></function_key>
    </sound>

    <sound>
      <name>ringout</name>
      <wav_file_name>full_ring.wav</wav_file_name>
      <attack_duration_time>0.010000000</attack_duration_time>
      <attack_level>1.00</attack_level>
      <decay_duration_time>0.000000000</decay_duration_time>
      <sustain_level>1.00</sustain_level>
      <release_start_time>0.000000000</release_start_time>
      <release_duration_time>∞</release_duration_time>
      <loop_from_time>0.000000000</loop_from_time>
```

```
      <loop_to_time>0.000000000</loop_to_time>
      <loop_limit>0</loop_limit>
      <start_time>2.995000000</start_time>
      <designer_volume_level>1.00</designer_volume_level>
      <designer_pan>0.00</designer_pan>
      <MIDI_program_number></MIDI_program_number>
      <MIDI_note_number></MIDI_note_number>
      <function_key></function_key>
    </sound>

    <routing></routing>
  </sounds>
</show_control>
```

The two sounds, ring and ringout, refer to the same WAV file, full_ring.wav, but in different ways. Ring plays the first 2.995 seconds of the file, then fades out to silence over the next 0.01 seconds. Ringout plays starting at 2.995 seconds, but it fades in from silence in the first 0.01 seconds. Notice thar the ringout sound has <release_duration_time></release_duration_time> set to infinity.

5.2 Theater_sound_sequence.xml

The file Theater_sound_sequence.xml looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<show_control>

  <!-- These sound sequence items are common to many shows in the theater.
       In the main sound sequence file we can offer telephone ring.
       It will appear on cluster 0.  -->

  <sound_sequence>
    <version>1.0</version>

    <sequence_item>
      <name>telephone-ring</name>
      <type>start_sound</type>
      <sound_name>ring</sound_name>
      <cluster_number>0</cluster_number>
      <tag>telephone-ring</tag>
      <text_to_display>Telephone ringing</text_to_display>
      <importance>2</importance>
      <next_release_started>telephone-ring-5</next_release_started>
      <next_sound_stopped>telephone-ring-7</next_sound_stopped>
    </sequence_item>

    <sequence_item>
      <name>telephone-ring-5</name>
      <type>start_sound</type>
      <sound_name>ringout</sound_name>
      <cluster_number>1</cluster_number>
      <tag>telephone-ring</tag>
      <text_to_display>Telephone ringout</text_to_display>
      <importance>2</importance>
      <next_completion>telephone-ring</next_completion>
```

```
      <next_termination>telephone-ring-8</next_termination>
    </sequence_item>

    <sequence_item>
      <name>telephone-ring-7</name>
      <type>start_sound</type>
      <sound_name>ringout</sound_name>
      <cluster_number>1</cluster_number>
      <tag>telephone-ring</tag>
      <text_to_display>Telephone ring end</text_to_display>
      <importance>2</importance>
    </sequence_item>

    <sequence_item>
      <name>telephone-ring-8</name>
      <type>wait</type>
      <time_to_wait>1</time_to_wait>
    </sequence_item>

  </sound_sequence>
</show_control>
```

When the operator rings the telephone it continues to ring until he tells it to stop, alternating between sounding the bell and letting the sound fade away, called “ringout”. When he presses the stop button the effect depends on whether the bell is ringing or fading out. If the bell is ringing it stops immediately and the audience hears the full ringout. If the ringout is already running it is allowed to continue to its end but the bell does not sound again. These sequence items permit the sound effects operator to make the sound of a ringing telephone. If the actor picks up the telephone handset during a ring, the ring stops, but it does not go silent immediately: it takes a few seconds to fade out.

The process starts with the telephone-ring sequence item. It starts playing the ring sound. If the sound is not stopped by the sound effects operator it perform the sequence item named in <next_release_started></next_release_started> when the release point of the ring sound is reached. That sequence item, telephone-ring-5, plays the ringout sound. Recall from above that the ring sounds ends with 0.01 seconds of fade out starting at 2.995 seconds, whereas the ringout sound starts with 0.01 seconds of fade in starting at 2.995 seconds. The effect then is to play the full_ring.wav file unmodified.

If the ringout sound is not stopped by the sound effects operator it executes telephone-ring, which plays the ring sound again. As before, if the sound effects operator does not stop the sound it executes telephone-ring-5. The result is that the audience hears the telephone bell every six seconds.

If the sound effects operator stops the sound between rings, the telephone_ring_5 sequence item transitions to telephone-ring-8 instead of telephone-ring when it is complete. Telephone-ring-8 just waits one second, so we don’t hear the ring again. If the sound effects operator stops the sound during a ring, sequence item telephone_ring immediately transitions to telephone_ring_7, which plays the ringout and terminates. The result is that the audience hears the ring sound stop short, but still hears the bell sound fade out to silence.

5.3 Example_03_sounds.xml

The file Example_03_sounds.xml has two sounds the background music and the automobile. We have a fancy theater with speakers at the back of the audience, so we run the background music through them.

```
<?xml version="1.0" encoding="utf-8"?>
<show_control>

  <!-- These are the special sounds used by Example_03.  -->
  <sounds>
    <version>1.0</version>

    <!-- The background sound is played from curtain open
       to curtain close.  -->
    <sound>
      <name>background music</name>
      <wav_file_name>background_music.wav</wav_file_name>
      <designer_volume_level>0.33</designer_volume_level>
      <designer_pan>0.00</designer_pan>
      <channels>
      <!-- Channel 0 to speaker rear_left, 1 to rear_right.  -->
      <channel>
        <number>0</number>
        <speakers>
          <speaker>
            <name>rear_left</name>
            <volume_level>1.0</volume_level>
          </speaker>
        </speakers>
      </channel>
      <channel>
        <number>1</number>
        <speakers>
          <speaker>
            <name>rear_right</name>
            <volume_level>1.0</volume_level>
          </speaker>
        </speakers>
      </channel>
      </channels>
    </sound>

    <sound>
      <name>Automobile</name>
      <wav_file_name>Automobile.wav</wav_file_name>
      <loop_from_time>16.578</loop_from_time>
      <loop_to_time>1.263</loop_to_time>
      <loop_limit>0</loop_limit>
      <release_duration_time>3.0</release_duration_time>
    </sound>

    <sound>
      <name>Automobile_end</name>
      <wav_file_name>Automobile.wav</wav_file_name>
      <start_time>16.578</start_time>
```

```
      <attack_duration_time>0.1</attack_duration_time>
    </sound>

  </sounds>
</show_control>
```

The music is played quietly because <designer_volume_level></designer_volume_level> is set to 0.33. The automobile sound is broken into three parts. From 0 to 1.263 seconds the motor starts. From 1.263 to 16.578 seconds the automobile is proceeding down the road. From 16.578 seconds to the end of the WAV file the automobile is pulling into a driveway and stopping. The amount of time needed for the automobile to reach the end of its journey is not fixed, so the traveling sound is played until the journey's end is signaled, at which time it fades out over three seconds, allowing the end of journey sound to be heard, which takes only 0.1 seconds to fade in.

5.4 Example_03_sound_sequence.xml

The file Example_03_sound_sequence.xml contains the special sound sequence for the production, and calls upon sequence items in Theater_sound_sequence.xml for the regular sounds made in this theater. It looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<show_control>

  <!-- This sound sequence is unique to Example_03.  These sequence items
       are available to the sound_effect_player's internal sequencer along
       with the sequence items in Theater_sound_sequence.xml.  -->

  <sound_sequence>
    <version>1.0</version>

    <sequence_item>
      <name>start_of_sequence</name>
      <type>start_sequence</type>
      <next>wait_for_curtain</next>
    </sequence_item>

    <sequence_item>
      <name>wait_for_curtain</name>
      <type>operator_wait</type>
      <text_to_display>Press Play when the curtain opens</text_to_display>
      <next_play>scene_01</next_play>
    </sequence_item>

    <sequence_item>
      <name>scene_01</name>
      <type>start_sound</type>
      <sound_name>background music</sound_name>
      <cluster_number>15</cluster_number>
      <text_to_display>background music</text_to_display>
      <tag>background music</tag>
      <next_starts>offer_telephone</next_starts>
    </sequence_item>
```

```
<sequence_item>
  <name>offer_telephone</name>
  <type>offer_sound</type>
  <next_to_start>telephone-ring</next_to_start>
  <cluster_number>0</cluster_number>
  <text_to_display>Telephone ring</text_to_display>
  <tag>telephone-ring</tag>
  <next>offer_automobile</next>
</sequence_item>

<sequence_item>
  <name>offer_automobile</name>
  <type>offer_sound</type>
  <next_to_start>Automobile_start</next_to_start>
  <cluster_number>1</cluster_number>
  <text_to_display>Automobile</text_to_display>
  <tag>Automobile</tag>
  <next>wait for end of scene 1</next>
</sequence_item>

<sequence_item>
  <name>Automobile_start</name>
  <type>start_sound</type>
  <sound_name>Automobile</sound_name>
  <cluster_number>1</cluster_number>
  <text_to_display>Automobile Running</text_to_display>
  <importance>2</importance>
  <tag>Automobile</tag>
  <next_sound_stopped>Automobile_end</next_sound_stopped>
</sequence_item>

<sequence_item>
  <name>Automobile_end</name>
  <type>start_sound</type>
  <sound_name>Automobile_end</sound_name>
  <next_to_start>Automobile</next_to_start>
  <cluster_number>1</cluster_number>
  <text_to_display>Automobile Stopping</text_to_display>
  <importance>2</importance>
  <tag>Automobile</tag>
</sequence_item>

<sequence_item>
  <name>wait for end of scene 1</name>
  <type>operator_wait</type>
  <text_to_display>Press Play when the curtain closes</text_to_display>
  <next_play>end of scene 1</next_play>
</sequence_item>

<sequence_item>
  <name>end of scene 1</name>
  <type>cease_offering_sound</type>
  <tag>Automobile</tag>
  <next>stop_automobile</next>
</sequence_item>
```

```
    <sequence_item>
      <name>stop_automobile</name>
      <type>stop_sound</type>
      <tag>Automobile</tag>
      <next>cease_telephone</next>
    </sequence_item>

    <sequence_item>
      <name>cease_telephone</name>
      <type>cease_offering_sound</type>
      <tag>telephone-ring</tag>
      <next>stop_telephone</next>
    </sequence_item>

    <sequence_item>
      <name>stop_telephone</name>
      <type>stop_sound</type>
      <tag>telephone-ring</tag>
      <next>stop_background</next>
    </sequence_item>

    <sequence_item>
      <name>stop_background</name>
      <type>stop_sound</type>
      <tag>background music</tag>
    </sequence_item>

  </sound_sequence>
</show_control>
```

When the Sound Effects Player is run, it looks for a start_of_sequence sequence item to tell it how to start. Here the start_of_sequence item executes wait_for_curtain. This is an operator_wait sequence item which tells the sound effects operator to press Play when the curtain opens, and when he does executes scene_01. Scene_01 plays the background music using cluster 15 and, when it starts playing, offers the telephone and automobile sounds before waiting for the end of scene 1.

During scene 1 the sound effects operator has two start buttons available, one for the telephone and the other for the automobile. The telephone sound was covered above. The automobile sound is an improvement on the car sound from example 2. When the sound effects operator starts the car the audience hears the motor start and the car running. The far running loop is long enough to sound natural. When the sound effects operator presses the Stop button the automobile pulls into the driveway and stops. The cross-fade between the car running and stopping is long to prevent an unnatural transition.

At the end of the scene the sound effects operator presses the Play button. All of the sounds are stopped and cease to be offered.

6 Sounds

These are the fields in a sound, with an description of each. The amplitude envelope uses attack, decay, sustain and release, abbreviated ADSR.

Field Name	description
name	The name of the sound, for reference from sequencer items.
wav_file_name	The name of the WAV file that contains the samples for this sound. This WAV file man be mono or stereo, 6000 to 96000 samples per second, 8 (S8 or U8), 16 (S16LE), 32 (S32LE or F32LE) or 64 (F64LE) bits per sample. If the file path is not absolute it is considered relative to the location of the file that references it. Internally the samples are converted to F32LE at 96000 samples per second.
attack_duration_time	The time to fade the sound in from silence, in seconds. Default is 0.0.
attack_level	The level that the sound will reach when the attack is complete. Default is 1.0, which is the full volume described in the WAV file.
decay_duration_time	The time to fade the sound from the attack peak down to the sustain level, in seconds. Default is 0.0.
sustain_level	The level to reach at the end of the decay. Default is 1.0, which is the full volume described in the WAV file.
release_start_time	The amount of time after which to start the release process, in seconds, which will trigger an action in the sequencer and begin to fade the sound out to silence. Default is 0.0, which means there is no time at which the release is automatically initiated, but it can still be initiated by the sound effects operator.
release_duration_time	The amount of time to fade the sound out to silence, in seconds. Default is 0.0, which stops the sound immediately. If the value is infinity ("∞") the sound does not fade at all.
loop_from_time	The time, in seconds, at which to stop playing the WAV file and start instead playing at the loop_to_time. Default is 0.0, which means not to do any looping.
loop_to_time	The time, in seconds, at which to start playing the WAV file after having

Field Name	description
	reached loop_from_time. Default is 0.0, which means play the WAV file from its beginning after hitting loop_from_time.
loop_limit	The number of times to pass through the loop. Default is 0, which means loop indefinitely. Looping will be terminated if the sound effects operator stops the sound.
max_duration_time	The maximum amount of time, in seconds, to load from the WAV file. This can be useful for infinite sources. Default is 0.0, which means no limit.
start_time	The time, in seconds, at which to start playing the WAV file. Default is 0.0, which means play the WAV file from its beginning.
designer_volume_level	WAV files are generally made as loud as possible without distortion, to minimize noise. This parameter lets you quiet the WAV file before it is mixed with other sounds. Default is 1.0, which performs no quieting.
designer_pan	For monophonic WAV files, positions the sound in the stereo field. For stereo WAV files, acts as a balance control, moving the sound towards the right or left speaker. -1.0 is full left, 0.0 is center, and +1.0 is full right.
default_volume_level	If you want to let the sound effects operator increase the volume of a sound after it starts playing, set this value to less than 1.0. He can increase it to 1.0.
MIDI_program_number	not implemented
MIDI_note_number	not implemented
function_key	not implemented
omit_panning	do not allow the sound effects operator to pan this sound. Default is False.

In addition, there are provisions for multi-channel sounds: a sound can have many channels and you can decide which speakers to send them to. See the background music in example 3 for details.

7 Sequence items

All sequence items have a name and a type. The name is used to refer to the sequence item from other sequence items. The type describes what each sequence item does. Each sequence item also has other fields, described below.

7.1 start_sequence

There may be only one of these in a sequence. When the Sound Effects Player starts, it looks for this sequence item to determine how to begin executing the sequence.

7.1.1 next

The sequence item to execute next.

7.2 start_sound

Start playing a sound, letting the sound effects operator control it from a specified cluster.

7.2.1 sound_name

The sound to play, as named in the in the Sounds section above.

7.2.2 cluster_number

Which cluster the sound effects operator will use to control this sound.

7.2.3 tag

A name by which to refer to this sound in stop_sound.

7.2.4 next_starts

The sequence item to execute when the sound starts to play.

7.2.5 next_completion

The sequence item to execute when the sound finishes playing and was not terminated early

7.2.6 next_termination

The sequence item to execute when the sound finishes playing and was terminated early.

7.2.7 next_sound_stopped

The sequence item to execute when the sound is stopped by the sound effects operator or some other external signal.

7.2.8 next_release_started

The sequence item to execute when the sound enters the release portion of its amplitude envelope. If the sound was terminated early this sequence item is not executed; instead the next_termination sequence item is executed when the sound is complete.

7.3 stop_sound

Stop a playing all sounds with the specified tag.

7.3.1 tag

The tag value of the playing sounds to stop.

7.3.2 next

When all sounds have been issued their stop commands, execute this sequence item.

7.4 wait

Wait for the specified number of seconds, displaying a message to the operator.

7.4.1 time_to_wait

The number of seconds to wait.

7.4.2 next_completion

The sequence item to execute when the wait is complete.

7.4.3 next

The sequence item to execute when the wait begins.

7.5 offer_sound

Allow the sound effects operator to play a sound by pressing the Start button on a cluster. The sound is controlled by the cluster. When it finishes the offer remains.

7.5.1 cluster

The number of the cluster that will contain this offer.

7.5.2 tag

The tag used in cease_offering_sound to remove this offer.

7.5.3 Q_number

The name used to activate this sound from another computer.

7.5.4 next_to_start

The sequence item to execute when the sound effects operator presses the Start button on the designated cluster.

7.5.5 text_to_display

The message to show the sound effects operator at the top of the designated cluster.

7.5.6 next

The sequence item to execute once the offer has been made.

7.6 cease_offering_sound

Stop allowing the sound effects operator to make a particular sound.

7.6.1 tag

The tag value of the offer_sound sequence items to cancel.

7.6.2 next

The sequence item to execute when the offerings have been removed.

7.7 operator_wait

Wait for the sound effects operator to press the Play button.

7.7.1 text_to_display

The message to display to the operator until he pushes the Play button.

7.7.2 next_play

The sequence item to execute when the operator pushes the Play button.

7.7.3 next

The sequence item to execute as soon as we start waiting.