



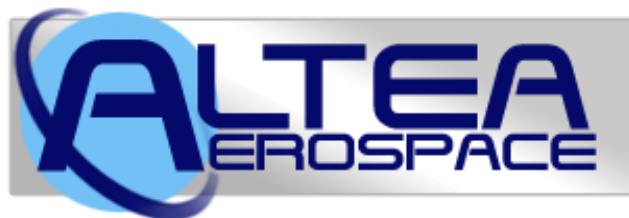
## User Manual

Version 1.0

*Publication Date: 20-February-2018*

*XRSound Version: 1.0*

XRSound is a product of Altea Aerospace.  
Powered by [irrKlang Pro](#) Sound Engine.



Copyright 2018 Douglas Beachy. All Rights Reserved.

*This software is freeware and may not be sold.*

Web: <http://www.alteaaerospace.com>

Email: <mailto:dbeachy@speakeasy.net>

Orbiter Forum: [dbeachy1](#) (<http://orbiter-forum.com>)

## Table of Contents

Copyright .....	2
Summary .....	3
XRSound Features .....	3
Requirements .....	5
Installation .....	6
Configuring XRSound .....	6
Added Custom Sounds to a Vessel via a Config File.....	6
Using the XRSound C++ APIs to Add Sounds to a Vessel.....	8



*The Future Is Now.*

## Copyright

This software copyright 2018 Douglas Beachy: this software is FREEWARE and may not be sold or distributed for a fee under any circumstances, including a "distribution fee." You may not redistribute this software or host it on a Web site; however, you are free to link to my Web page at <http://www.alteaaerospace.com>, and to link with XRSound.lib in own projects and/or create custom XRSound-<vessel class>.cfg files to add any of the included default sounds (and, of course, any new sounds you create) to your own vessels. However, you may not redistribute any of the default sounds.

You may not charge a fee of any kind to use this software, nor may you use this software for any commercial purpose (i.e., where profit is involved), regardless of the terms governing the Orbiter instance on which it is running, without express written permission signed by the author and sent via hardcopy letter or fax; i.e., an email is not sufficient to grant permission. Please email me at [dbeachy@speakeasy.net](mailto:dbeachy@speakeasy.net) if you have any questions.

This software is provided without any warranty, either expressed or implied.

## Summary

XRSound provides over 450 available, fully configurable, default sounds and voice callouts for all Orbiter vessels. In addition, it provides a C++ API for add-on Orbiter vessels authors to use via the XRSound.dll Orbiter plugin module.

XRSound is fully configurable, and in addition to the global `XRSound.cfg` file that configures default sounds for all vessels, you can also edit or create Orbiter vessel-class-specific config files, such as the included `XRSound-DeltaGlider.cfg`, etc., that assigns default sounds to vessel-specific animations and landing gear without requiring recompiling the vessel or doing any coding.

Special thanks to Grzegorz Lorens ([Loru](#) on Orbiter-Forum) for creating the XRSound logo and a brand-new ambient music track that by default plays in external views in space (details below).

## XRSound Features

- **Automatically adds default sounds for any Orbiter vessel that defines thrusters -- not just XR vessels.** You can also manually assign sounds to any vessel class via its `XRSound-<vessel class>.cfg` file; e.g., `XRSound-DeltaGlider.cfg`.
- Each sound or group of sounds is fully configurable via `XRSound.cfg` and optional Orbiter vessel class-specific config files, or via the XRSound SDK (included).
- Includes over 450 default sound files, including all [XR vessel](#) voice callouts by actress [Sally Beaumont](#) and 141 ATC callouts from STS-114 and STS-121 featuring CAPCOM [Julie Payette](#). These callouts were recorded live from NASA TV and saved as 48KHz, 16-bit mono.
- Add-on authors are free to use any of the sounds in their own vessels (just please don't redistribute the files).
- Includes new *Solar Serenity* ambient music track created by and licensed from Grzegorz Lorens ("[Loru](#)" on Orbiter-Forum) that plays in external views in space by default. This is configurable like any other sound. (For a lossless version of Loru's *Solar Serenity*, take a look [here](#).)
- All default sounds are fully configurable, including:
  - Air conditioning
  - Audio greeting voice callout on simulation start
  - Music in external views in space
  - Wind effects while landed in an atmosphere
  - Main engine sound
  - Hover engine sound
  - Retro engine sound

- RCS sounds
  - Switch on / Switch off clicks for RCS and AF Ctrl changes
  - RCS mode change voice callouts ("Rotation", "Translation", "Off")
  - AF Ctrl mode voice callouts ("Off", "Pitch", "On")
  - Crash sound
  - Hard landing sound
  - Tire chirp on touchdown for vessels that define landing gear animation ID via their `XRSound-<class name>.cfg` file. [You can set ``LogVesselAnimations = 1`` in `XRSound.cfg` to log the animation IDs for all vessels as you activate each animation.]
  - Tires rolling sound; volume varies by ship velocity
  - One additional custom engine sound per vessel (e.g., the SCRAM engines in the stock DeltaGlider-S). Configurable via a given vessel class's `XRSound-<vessel class>.cfg` file.
  - Wheel brakes sound
  - Takeoff and landing voice callouts ("100 knots", "Wheels up" / "Liftoff", "You are cleared to land", "Warning: gear is up", "Touchdown", "Wheel stop", etc.)
  - Wind sound and plasma sound in flight; volume varies by dynamic pressure and distance
  - Autopilot on / off tones
  - Ambient sounds configurable by pointing to a folder in `XRSound.cfg`; twelve ambient sounds are included by default, but there is no limit to the number of ambient sounds supported. Minimum and maximum playback interval is configurable.
  - Fully configurable music folder support; music files may be played back sequentially or at random (or be disabled), and may be configured to play internally / externally / in space / while landed, or any combination thereof.
  - Altitude voice callouts
  - Docking distance voice callouts
  - Docking and undocking voice callouts and sounds
  - Docking radar sounds whose beep interval varies by the distance to the docking port
  - Mach voice callouts ("Mach 1", "Mach 2", "Mach 27 Plus", "Subsonic", etc.)
  - ATC sounds played at random at configurable intervals from a specific folder. Like all other sounds and groups, the folder can be changed at runtime via an SDK call and may be configured via `XRSound.cfg` or `XRSound-<vessel class>.cfg` otherwise.
- The new [XR vessel versions](#) use `XRSound`, and therefore no longer need to bundle their sound files with each vessel anymore: all sounds are included with the `XRSound` download.
  - Can add sound events to any existing Orbiter vessel's animations simply by editing that vessel's class's `XRSound-<classname>.cfg` file. Includes `.cfg` files for each of the default Orbiter vessels.
  - Includes C++ SDK for add-on authors to use in their own vessels; the new XR vessel versions use it.

- You can replace or disable any or all of the default sounds via the XRSound SDK or XRSound.cfg.
- XRSound.lib is statically linked; add-on vessel authors using XRSound do not need to bundle Visual Studio redistribution files with their add-on in order to use XRSound, and add-ons that link with XRSound.lib can still run without XRSound.dll installed.
- No limit to the number of sound slots that vessels may use.
- No limit to the number of vessels in the simulation.
- No limit to the number of ATC radio chatter files for a given "frequency" (i.e., folder): just configure the ATC folder in your XRSound.cfg, and any sound files in that folder are played at random for ATC chatter.
- No limit to the number of music files in the configured music folder.
- Parameters for minimum and maximum times between ATC chatter playback are configurable via XRSound.cfg.
- Uses [irrKlang Pro](#) sound engine internally, which supports the following sound file formats:
  - RIFF WAVE (\*.wav)
  - Ogg Vorbis (\*.ogg)
  - MPEG-1 Audio Layer 3 (\*.mp3) [*via ikpMP3.dll plugin, included*]
  - Free Lossless Audio Codec (\*.flac) [*via ikpFlac.dll plugin, included*]
  - Amiga Modules (\*.mod)
  - Impulse Tracker (\*.it)
  - Screamer Tracker 3 (\*.s3d)
  - Fast Tracker 2 (\*.xm)
  - Additional sound formats may be supported by dropping irrKlang plugin DLLs into \$ORBITER\_ROOT.
- Refer to the comments in \$ORBITER\_ROOT\XRSound\XRSound\*.cfg files for full details about configuring XRSound.
- Refer to the comments in XRSound.h (used with static library XRSound.lib or XRSoundD.lib) in \$ORBITER\_ROOT\Orbitersdk\XRSound for more information about the XRSound API.

## Requirements

- Windows Vista or newer.
- [Orbiter 2016](#). Older Orbiter releases are not compatible with XRSound.

## Installation

This section details how to install and activate XRSound vessel. Note that XRSound is a plug-in for *Orbiter*, and requires that [Orbiter 2016](#) be installed first.

1. Install [Orbiter 2016](#). Older versions of Orbiter are NOT SUPPORTED by XRSound.
2. Unzip the XRSound distribution file into your `$ORBITER_ROOT` (e.g., `C:\Orbiter`) directory.
3. Bring up Orbiter to display the Orbiter Launchpad.
4. Click the *Modules* button.
5. Click the checkbox next to *XRSound* in the *Sound module for Orbiter* section to activate XRSound.

*Note: if you previously installed the OrbiterSound add-on, which does not fully support Orbiter 2016, deactivate it by unchecking it in the same section. Otherwise, its default vessel sounds will conflict with XRSound's.*

Now boot up Orbiter and load one any of the launch scenarios; you should hear an audio greeting voice callout when the simulation starts. If you don't, look at `$ORBITER_ROOT\XRSound.log` for error or warning messages.

---

## Configuring XRSound

XRSound is fully configurable and allows users to replace or disable any sound or voice callout either globally, via the `$ORBITER_ROOT\XRSound\XRSound.cfg` file, or on a per-vessel-class basis via `$ORBITER_ROOT\XRSound\XRSound-<vessel class name>.cfg` file (for example `C:\Orbiter\XRSound-DeltaGlider.cfg`). Settings in the vessel class's .cfg file override any corresponding settings in the global `XRSound.cfg` file.

Refer to the comments in `$ORBITER_ROOT\XRSound\XRSound.cfg` for details about how to configure XRSound.

## Added Custom Sounds to a Vessel via a Config File

There are two ways to define custom sounds and/or voice callouts to an existing vessel. The first is so use the XRSound C++ API and make XRSound calls from the vessel's source code. This allows you full access to all of XRSound's features, but also requires you to have access to the vessel's source code.

The second way is to create or edit an `$ORBITER_ROOT\XRSound\XRSound-<className>.cfg` file for the vessel's class in Orbiter. In addition to defining (or "overriding") any default sounds defined in `XRSound.cfg`, you can also define new sounds for any the vessel's animations (for example, a nosecone opening or closing), as well as define one custom engine sound (for example, SCRAMJETS on the default DeltaGlider-S. `XRSound` includes custom `.cfg` files for each of Orbiter's default vessels. For example, the DeltaGlider-S's Orbiter class name is "DG-S", and so its custom `XRSound.cfg` file is named `XRSound-DG-S.cfg`.

*Note: if a vessel's Orbiter class name contains any characters that are invalid filename characters (e.g., "/"), they will be converted to underscores. For example, a vessel class named `Foo/BarClass` becomes `Foo_BarClass`, so its `XRSound` config file would be `XRSound-Foo_BarClass.cfg`.*

Here is how you would typically go about adding some new sounds to an existing vessel via its `XRSound` config file:

1. Edit `$ORBITER_ROOT\XRSound\XRSound.cfg` and make these edits:

```
EnableVerboseLogging = 1
LogVesselAnimations = 1
LogThrusterData = 1
```

Note: These settings should only be enabled temporarily while you are doing your `XRSound` development, because having that extra logging enabled can greatly increase the rate at which the `$ORBITER_ROOT\XRsound.log` file grows!

2. Run Orbiter and launch a scenario with the vessel whose sounds you want to customize.
3. Activate the vessel animation you want to define sounds for (for example, deploy the landing gear with the "G" key).
4. Look in `XRSound.log` for lines like this:

```
[GL-01] >> LogVesselAnimations: [DeltaGlider][animation
ID = 7, state = Moving]
```

That line shows that the landing gear animation for the vessel named `GL-01` with Orbiter vessel class `DeltaGlider` has the Orbiter-assigned animation ID of 7.

5. Since the vessel class is `DeltaGilder`, the file you need to edit to assign custom sounds to it is `$ORBITER_ROOT\XRSound\XRSound-DeltaGlider.cfg`. Open that file in your favorite text editor. (If the file does not exist yet, copy an existing vessel's `cfg` file as a template and then delete any existing settings in it before defining new ones.)
6. Add or edit sounds for the animation events for animation ID you are interested in (e.g., ID 7 for the DeltaGlider's landing gear). Refer to the detailed comments in `$ORBITER_ROOT\XRSound\XRSound-DeltaGlider.cfg` for details about when each custom animation sound and/or thruster sound plays.

7. Don't forget to reset your logging settings back to zero in `$ORBITER_ROOT\XRSound\XRSound.cfg` once you are done adding and debugging your sounds:

```
EnableVerboseLogging = 0
LogVesselAnimations = 0
LogThrusterData = 0
```

## Using the XRSound C++ APIs to Add Sounds to a Vessel

XRSound includes a simple, easy-to-use API to play and manipulate sounds from a vessel. These SDK files are installed to `$ORBITER_ROOT\Orbitersdk\XRSound`:

<code>XRSound.h</code>	Defines and documents the API methods
<code>XRSound.lib</code>	Release-mode static C++ library
<code>XRSoundD.lib</code>	Debug-mode static C++ library

For Debug builds, you should link with `XRSoundD.lib`. Link with `XRSound.lib` for Release builds. The XRSound static libraries were compiled using Visual Studio 2015, so using Visual Studio version 2015 or later is recommended; earlier Visual Studio versions have not been tested. Note that these libraries are lightweight and simply dynamically invoke the corresponding methods in `XRSound.dll`, if present.

For full details about the XRSound C++ API methods, refer to the detailed comments in `$ORBITER_ROOT\Orbitersdk\XRSound\XRSound.h`. Here are some simple examples of how to use the XRSound API:

```
include "XRSound.h"

// could use #define for your custom sound IDs instead since sound IDs
// are just signed integers, but an enum is cleaner
enum MySounds
{
    SystemReset,    // value 0
    SomeOtherSound,
    ...
};

class MyVessel : public VESSEL4
{
    ...
    XRSound *m_pXRSound;
    bool bResettingTheSystem;
};

void MyVessel::clbkPostCreation()
{
    m_pXRSound = XRSound::CreateInstance(this);    // create sound
    engine instance for this vessel
}
```



```

        // load a custom sound for this vessel
        m_pXRSound->LoadWav(SystemReset, "XRSound\Default\System
Reset.wav", XRSound::Radio); // returns false if file not found

        // disable the default "100 knots" voice callout
        m_pXRSound->SetDefaultSoundEnabled(XRSound::OneHundredKnots,
false);

        // replace the default docking voice callout with a custom one that
resides in $ORBITER_ROOT\MySoundsFolder
        m_pXRSound->LoadWav(XRSound::DockingCallout, "MySoundsFolder\My
custom docking callout.mp3", XRSound::Radio);
    }

void MyVessel::clbkPreStep(double simt, double simdt, double mjd)
{
    if (m_bResettingTheSystem)
    {
        // play our custom sound: don't loop it, and use max volume
        m_pXRSound->PlayWav(SystemReset, false, 1.0); // returns
false if play fails
        m_bResettingTheSystem = false; // so we don't keep looping
it (although calling PlayWav if the sound is already playing can only
change its volume or loop settings: it will not stop and restart it.
    }
}

MyVessel::~MyVessel()
{
    // as with any other allocated member variables, always remember to
clean up the XRSound engine instance in your vessel's destructor! :)
    delete m_pXRSound;
}

```

Of course, you can also load a new sound into the same sound slot at any time; if a sound is already playing in that slot, it will automatically be stopped when LoadWav replaces it:

```

if (m_pXRSound->LoadWav(SystemReset, "My new sound.ogg",
XRSound::Radio)) // does sound file exist?
    m_pXRSound->PlayWav(SystemReset); // play the sound using
defaults of loop = false and max volume

```

One important thing to note regarding performance is that LoadWav is lightweight and does not actually *load* the sound data into memory: it simply 1) verifies that the sound file exists and is readable, and 2) saves the file path and playback type in the master sound map in memory for this vessel. Sound data is only loaded when it is played, and it automatically released when no longer needed: this is all managed by the underlying [irrKlang Pro](#) sound engine.

Note that XRSound does not write any data to or read any data from Orbiter scenario files.

Refer to `$ORBITER_ROOT\Orbitersdk\XRSound\XRSound.h` for more information about the XRSound SDK. For questions or support regarding XRSound, visit <http://www.orbiter-forum.com>.

-- end --