oxid

Paraphonic String Synthesizer
Version 1.0

www.fullbucket.de/music

Table of Contents

Introduction

Oxid is a software instrument plug-in for Microsoft Windows (VST2/VST3/CLAP) and Apple macOS (VST2/VST3/AU/CLAP) simulating the classic ARP *Omni-2 Polyphonic Synthesizer* from the 1970s. It is written in native C++ code for high performance even on "lighter" systems. The main features are:

- Close simulation of the original hardware
- 64 voices "polyphony"
- *String* and *Synthesizer* sections
- Paraphonic madness
- Additional Phaser and Delay effects
- All parameters can be controlled by MIDI controllers
- Plug-in supports Windows and macOS (32 bit and 64 bit)

Oxid is based on the **iPlug2** framework maintained by **Oli Larkin and the iPlug2 team**. Big thanks, guys!!! Without your work it would not have been possible to create a resizable *Oxid* user interface.

To resize the plug-in you just grab the yellow triangle at the bottom right of the window and drag it. You can save the current window size using the menu entry "Save Window Size" in the *Options Menu* or by right-clicking somewhere into an empty space of *Oxid*'s panel.

If you have trouble with the standard version of *Oxid*, please grab the (sound-wise identical) "N" version of the plug-in which is based on the original **iPlug** framework.

Acknowledgments

- **Oli Larkin** and the **iPlug2** team.
- **kraftraum** (https://soundcloud.com/kraftraum) has designed the default programs 32 – 62 and did the Beta testing.
- **Keith Robert Murray** aka *Synthesizer Keith* (www.keithrobertmurray.com) and **Marc Doty** aka AutomaticGainsay for their very helpful YouTube videos.
- **Nick Cursley** who asked me to do an ARP *Omni* simulation.

Encore: Omni

The first time I heard (about) the ARP *Omni* was on Tangerine Dream's live album *Encore* – it is listed on the back of the double LP as part of Edgar Froese's gear. Of course many other great artists played it too, and somewhere on the internet someone even claims that "the *Omni* was the *DX-7* of the 70s". Well, you immediately recognize its sound when you hear it... but wait a minute: What you hear might also come from a *Solina* or any other string machine of that time, right[1]? One really has to be an expert to distinguish between those instruments because they all kind of sound the same. The emphasis is on "kind of": It is the *Ensemble* effect that almost all string machines rely on. More on that later.

If it were not for the term *String Synthesizer*, the *Omni* just seemed to be a successor to the famous *Solina String Ensemble*. But ARP added an additional VCF, VCA and ADSR envelope[2] to the existing organ-like design (plus a simple monophonic Bass section). While the sound generation was still fully polyphonic, the synthesizer section now was *paraphonic* – all voices go through a single filter and amplifier. Anyway, this still was something in 1975 (and not much different from the design of other "multi keyboards" of that area)!

In 1977 the *Omni-2* followed with some slight improvements and the new classic ARP orange/black color scheme. But like its competitors, it started to face stiff competition from "true polyphonic" synthesizers like the *Prophet 5* (1978) or the *OB-X* (1979), and even its successor, the *Quadra*, couldn't catch up anymore.

Throughout this manual the name "*Omni*" refers to both the ARP *Omni* and the ARP *Omni-2* models, unless otherwise noted.

The Subject

If you look at my plug-in portfolio you will see that I already have simulated a lot of string machines. So why one more? Because this one is (like all the others) a little bit different. It is amazing to see all the various designs, and they are nice challenges for a nerdy developer like me.

In case of the *Omni* it starts with the tone generator: The square wave signals derived from the single top-octave synthesizer and the subsequent octave dividers are shaped into sawtooth-like signals using a simple diode/capacitor/resistor circuitry. However, the resulting signals are not sawtooth waves but something like exponential "ramps". Via the funky "HOLLOW WAVEFORM" switch these "ramps" are transformed into alternating positive/negative exponential "ramps" that sound almost like square waves[3].

After passing a formant filter network, the tone generator signals are used for both the *String* and the *Synthesizer* sections. While the *String* signals are fed directly into the *Chorus Phaser* section (i.e. the Ensemble effect; note that there is no Phaser at all), the *Synthesizer* signals run through the single VCF and VCA. The monophonic

1 A bold statement, I know.

2 VCF: Voltage Controlled Filter
VCA: Voltage Controlled Amplifier
ADSR: Attack/Decay/Sustain/Release – an envelope generator

3 Surprisingly this is "less expensive" than using the original square waves here.

String Bass/Cello as well as the *Synth Bass* sections are derived from the octave divider signals, too, and share their own "wave shaper" circuitry.

The Pitfalls

So far everything is simple and crystal clear. But now it gets a bit messy: With the "RELEASE" slider you can set the release time of each note – fully polyphonically, because the gating circuits provide the release function for each key. But the single ADSR envelope controls the amplitude contour of the *Synthesizer* section as well, and this can have surprising results. For example, when you set a short "RELEASE" time in the *String* section, you will not get a longer release time with the ADSR in the *Synthesizer* section (because both sections use the same tone generator signals). And if you set a long "RELEASE" time and play some melody, you might still hear the released notes in the *Synthesizer* section – even for short release times of the ADSR.

Next thing is the "ATTACK" slider which controls the attack time of the *String* section (only!). The attack here behaves like the *Crescendo* on the *Solina* and works paraphonically: When you press a key and then another one *before* the attack phase has finished, the envelope restarts the attack phase *for all played notes* (!). However, when you press a key *after* the attack phase has finished, the new note will not have *any attack at all* and starts immediately.

To make things even more complicated, you can send the output of the *Synthesizer* section into the *Chorus Phaser* section of the *String* section. A nice feature, but keep in mind that the chorused *Synthesizer* signal is now also sent through the *String* sections attack circuitry (see above) and the respective *String* section "MIX" control. I always have to think twice when I program a chorused synth sound and wonder why it does not work quite as expected.

You want some more? OK: The *String Bass/Cello* as well as the *Synth Bass* sounds only play the lower notes below or equal to G2 (MIDI note number 55). But when at least one of the *Synth Bass* sounds is selected, the *Synthesizer* section only plays the upper notes higher than G2 ("split mode"). This is not true for the *String* section nor when one or both *String Bass/Cello* sounds (but none of the *Synth Bass* sounds) are selected. Now when you use the chorused synthesizer signal and select a *Synth Bass* sound, the *String* section's paraphonic attack will be activated even if you press a bass note (below or equal to G2) where the *Synthesizer* section is not playing. ☺

Chorus Phaser – The Ensemble Effect

The *Omni*, like the *Solina*, features three modulated delay lines in the *Chorus Phaser* section to create something which is actually not a Phaser but a triple Chorus – the classic Ensemble effect. Unlike the *Solina,* the *Omni* uses three (instead of two) independent LFOs to modulate the delay times (one LFO per delay line), and the dry signal is not mixed to the output. The resulting effect is a little bit different but definitely produces "the" string machine vibe.

As stated in section *The Pitfalls* you can route the *Synthesizer* signal to the *Chorus Phaser* section, but for some reason the ARP engineers decided that activating this function automatically sets the three LFOs to a much lower modulation rate (still not a Phaser though). Consequently the chorused synthesizer signal always goes with a "slow" Ensemble effect which applies to the *String* section as well[4].

4 With *Oxid* you can (re)set the modulation rates from "slow" to "fast".

The Summary

Developing *Oxid* as a simulation of the ARP *Omni-2* was fun. Of course the instruments lacks many features like “real” polyphony, velocity, after touch, or even VCF keyboard tracking which nowadays are considered to be standard. But hey: The original *Omni* was developed in 1975 (the *Omni-2* followed in 1977) and things had to evolve. All the quirks caused by the design and especially its restrictions can have an inspirational effect on the music you make – or the opposite.

I will not argue whether *Oxid* sounds exactly the same as the *Omni*. I did the best I could given that I am not in possession of the original hardware itself. But as always, I am open for your comments on *Oxid*, though. ☺

Oxid Operation

Oxid basically consists of three main sections: *String*, *Synthesizer* and (*Synth*) *Bass*. Additionally, there is a phaser and a delay effect not available on the original *Omni* hardware. The instrument itself is 64-voice polyphonic.

String Section

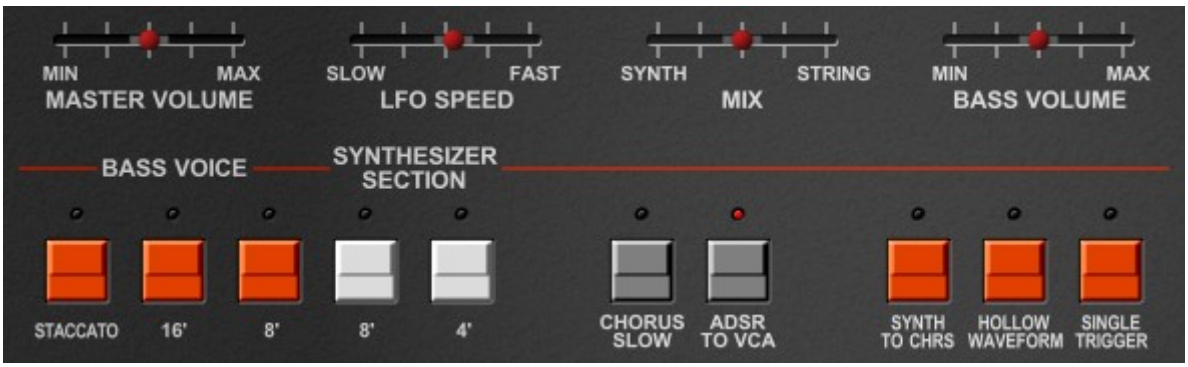
The *String* section features two polyphonic presets: *Violin* (4') and *Viola* (8'), a one octave lower and a little bit "duller" variation of *Violin*. *Bass* (16') and *Cello* (8') are monophonic presets that only play on keys lower or equal to G2 (MIDI note #55).

The "TONE" control shifts the frequencies of the internal formant filters for a "brighter" or "duller" sound of the instrument. Note that this control is not available on the original *Omni* and applies for all sections (*String*, *Synthesizer*, *Synth Bass*).

"ATTACK" and "RELEASE" control the respective timing of the *String* section's envelope generator. As explained in section *The Pitfalls*, the attack works paraphonically and restarts for all played notes until the attack phase is finally finished. In contrast to this, the release works polyphonically. Note that the *String Bass/Cello* presets are not affected by the "ATTACK" and "RELEASE" controls and have their own fixed envelope.

Synthesizer, *Synth Bass* And *Mix* Section

The layout of the *Omni* (and *Oxid*) is a bit confusing in terms of separating these three sections. The *Synthesizer* section (like the *String* section) has two polyphonic presets *4'* and *8'*, and the *Synth Bass* section two monophonic presets *8'* and *16'*.



The *Synthesizer*'s VCF and ADSR are explained later. The *Synth Bass* section's 2-pole VCF and its envelopes are set to fixed values – the only thing you can modify here is the decay to a shorter time via the "STACCATO" switch. Once a *Synth Bass* preset is selected, the *Synth Bass* only plays notes below or equal G2 (MIDI note #55) while the *Synthesizer* only plays notes above G2.

Some more controls:

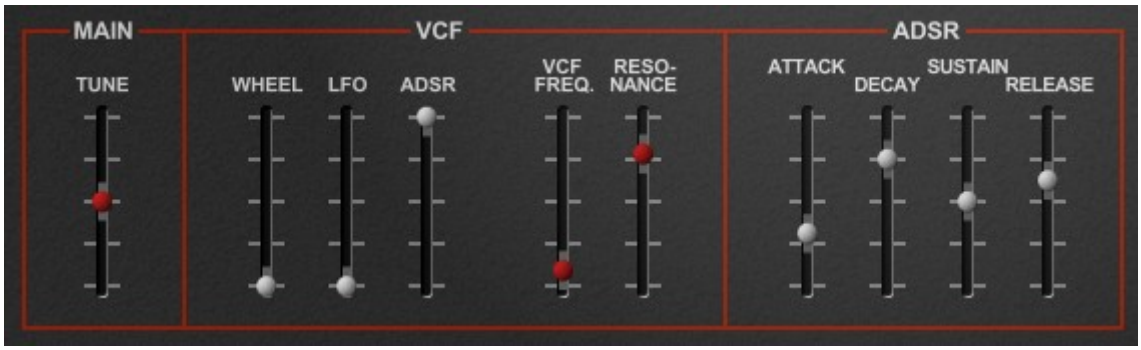
- "MASTER VOLUME" controls exactly that.
- "LFO SPEED" sets the rate of the LFO modulating the *Synthesizer*'s VCF.
- "MIX" controls the balance between the *Synthesizer* and the *String* section. Note that the chorused *Synthesizer* signal appears on the *String* section only!
- "BASS VOLUME" sets the volume of the *String Bass/Cello* and the *Synth Bass*.
- The "SINGLE TRIGGER" switch activates single-triggering of the ADSR envelope (the *Synth Bass* section always uses single trigger which cannot be changed).
- "HOLLOW WAVEFORM" changes the sawtooth-like waveform of the *Synthesizer* and *String* sections (but not the *String/Synth Bass*!) to a square-like waveform.
- "SYNTH TO CHRS" sends the *Synthesizer* signal to the *Chorus Phaser* section and from there to the *String* section's attack circuit, mix and final output. It also activates the lower modulation rates of the Ensemble effect. Note that on the *Omni* the original label for this button is "CHORUS PHASER".

The following two controls do not exist on the original *Omni*:

- "CHORUS SLOW" toggles between the lower and higher modulation rates of the Ensemble effect. On the *Omni*, the chorused *Synthesizer* signal is always processed using the lower rates which is an unnecessary restriction.
- "ADSR TO VCA" enables the ADSR generator to control the amplitude contour of the *Synthesizer* section; this is the default and the way it works on the original *Omni*. However, when using the ADSR for VCF modulation purposes, it can be handy to deactivate the "ADSR TO VCA" control. In this case, the amplitude contour is the same as in the *String* section but without the attack phase.

Synthesizer Section (Continued)

The *Synthesizer* section features some more controls for the VCF and ADSR as well as the overall tuning.



In the subsection "MAIN", the slider "TUNE" controls the master tune of the instrument. Note that this refers to all three sections *Synthesizer*, *String*, and *Synth Bass*.

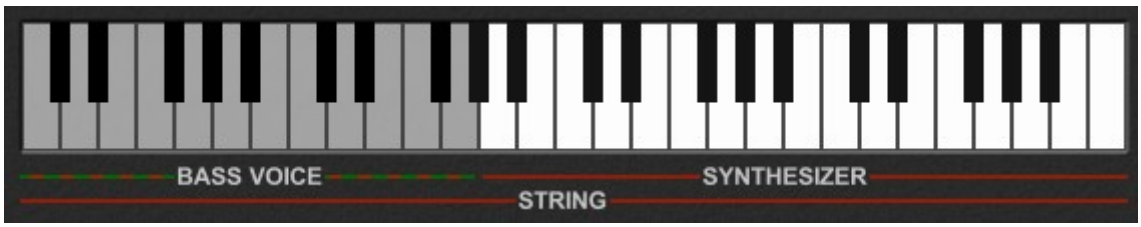
The subsection "VCF" contains five sliders; the first three determine the amount of VCF cutoff frequency modulation for the Modulation Wheel, the LFO and the ADSR

generator, while the last two sliders set the VCF's cutoff frequency and its resonance. Note that the VCF of the *Omni* (and *Oxid*) is not able to self-resonate.

Finally, the subsection "ADSR" contains the four sliders to set Attack, Decay and Release time, and of course the Sustain level.

Keyboard Section

Usually I do not add virtual keyboards onto the user interfaces of my plug-ins, but since *Oxid* features some keyboard splitting when the (*Synth*) *Bass* is active, I thought it might be useful to have one.



The keyboard shows the 49 keys that are available on the original *Omni* (however, *Oxid* can play even more notes below and above the visible range). You can click any of these keys to play a note; right-clicking "holds" a note until you click the key again.

The gray range of the keyboard represents the notes that work for the *String/Synth Bass* presets. If a *Synth Bass* preset is selected, the white keys denote the notes that can be played using the *Synthesizer* presets. *String* presets ignore any key split and always work for the whole keyboard.

Panorama Section

The *Omni* has three mono output jacks for the three sound sections while *Oxid* provides a single stereo output and a panorama mixer. Thus you can set the stereo position for each section separately.

Phaser And Delay

The most common effects used along with string machines are phasers and delays (think *Oxygène* and *Equinoxe…*). Since the *Omni* does not include a dedicated effect section[5], I decided to add these two effects for your convenience.

The Phaser is loosely based on the famous Electro Harmonix *Small Stone* with its 4-stage allpass filter design. "SPEED" sets the modulation rate and "FEEDBACK" the positive or negative feedback level – extreme settings can make your ears tingle. The Phaser works in stereo and can be activated for each of the three sound sections individually.

The Delay, like the *Chorus Phaser*, is modeled after the classic Bucket Brigade Devices (BBD) that popped up in the early 70s. It works in stereo on the output of the Phaser section.

5 It took some years before adding effect sections to synthesizers became a standard.

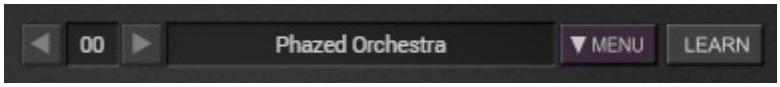
You can set the delay time from 75ms to 600ms or, when the button "×2" is active, from 150ms to 1200ms. "FEEDBACK" controls the amount of the delay output signal that is fed back into the input, and "MIX" controls the balance between the original and the effect signal.

"SYNC" enables the synchronization of the delay time to the host's tempo. "VIBE" adds some slight modulation of the delay time to produce bright stereo effects, and "P/P" activates the Ping-Pong mode where the output of the left delay channel is fed into the input of the right channel and vie versa.

Plugin Handling

Control Section

All settings of *Oxid* can be saved as a named program. You can also select one of the 64 programs via the arrow buttons or by clicking on the program number.



Options Menu

When clicking the *Menu* button in the *Control* section, a context menu opens with the following options:

Copy Program	Copy current program to internal clipboard
Paste Program	Paste internal clipboard to current program
Init Program	Initialize the current program
Load Program	Load a program file containing a patch to *Oxid's* current program
Save Program	Save *Oxid's* current program to a program file
Load Bank	Load a bank file containing 64 patches into *Oxid*
Save Bank	Save *Oxid's* 64 patches to a bank file
Select Startup Bank	Select the bank file that should always be loaded when *Oxid* is started
Load Startup Bank	Load the Startup bank file; can also be used to check what the current Startup bank is
Unselect Startup Bank	Unselect the current Startup bank
Default Path for Program Files	Sets the default path for program and bank files
MIDI Thru	Set globally if MIDI data sent to *Oxid* should be sent through to its MIDI output (stored in configuration file)
Ignore Program Change	Set globally if MIDI Program Change data sent to *Oxid* should be ignored (stored in configuration file)
Reload Configuration	Reload *Oxid's* configuration file
Save Configuration	Save *Oxid's* configuration file
Check Online for Update	When connected to the Internet, this function will check if a newer version of *Oxid* is available at fullbucket.de
Visit fullbucket.de	Open fullbucket.de in your standard browser

The *oxid.ini* Configuration File

Oxid is able to read some settings from a configuration file (`oxid.ini`). The exact location of this file depends on your operating system and will be displayed when you click on “Reload” or “Save Configuration”.

MIDI Control Change Messages

All parameters of *Oxid* can be controlled by MIDI controllers, or more precise: Each MIDI controller (except *Modulation Wheel* and *Sustain Pedal*) can control one of *Oxid*'s parameters. The mapping is defined in the `oxid.ini` for example like this:

```
[MIDI Control]
CC7  = 0  # Master Volume
CC70 = 29 # VCF Cutoff
CC71 = 30 # VCF Resonance
...
```

The syntax is straight forward:

```
CC<controller number> = <parameter ID>
```

Given the above example, controller 7 directly controls the *Master Volume* parameter, controller 70 the *VCF Cutoff* etc. As you can see, comments are introduced by the Pound sign (#); they are here just for description purposes and completely optional. Note that the *controller number* can run from 0 to 110, with the exception of 1 (*Modulation Wheel*) and 64 (*Sustain Pedal*); the latter two are simply ignored.

MIDI Learn

The easiest way to assign MIDI controllers to *Oxid* parameters is to use the *MIDI Learn* function. To activate MIDI Learn, click on the LEARN button and wiggle both the MIDI controller and the *Oxid's* parameter that you want to link. If you want to unlearn the assignment, right-click the LEARN button (the label now reads “UNLEARN”) and activate it. Now wiggle the MIDI controller or the parameter that you want to unlearn.

Parameters

General

parameter	id	description
Master Volume	0	Master volume
Tune	1	Master tune
Tone	2	Tone (brightness)
Attack	3	Attack time of the *String* section
Release	4	Release time of the *String* section
String: Bass	5	Activates the *String Bass* preset
String: Cello	6	Activates the *String Cello* preset
String: Viola	7	Activates the *String Viola* preset
String: Violin	8	Activates the *String Violin* preset
LFO Speed	9	Speed of the LFO modulating the *Synth* VCF
Mix String/Synth	10	Balance between *String* and *Synth* section
Bass Volume	11	Volume of the *String Bass/Cello* and *Synth Bass*
Bass: Staccato	12	Activates the Staccato setting (shorter decay time) of the *Synth Bass*
Bass: 16'	13	Activates the *Synth Bass 16'* preset
Bass: 8'	14	Activates the *Synth Bass 8'* preset
Synth: 8'	15	Activates the *Synth 8'* preset
Synth: 4'	16	Activates the *Synth 4'* preset
Chorus: Synth	17	Routes the *Synth* to the *Chorus Phaser*
Hollow Waveform	18	Activates the "hollow" (square-like) waveform
Single Trigger	19	Activates the Single Trigger for the *Synth* section
VCF Wheel	20	Amount of VCF modulation via Modulation Wheel
VCF LFO	21	Amount of VCF modulation via LFO
VCF ADSR	22	Amount of VCF modulation via ADSR
VCF Cutoff	23	VCF cutoff frequency
VCF Resonance	24	VCF resonance
Synth: Attack	25	ADSR Attack time
Synth: Decay	26	ADSR Decay time

parameter	id	description
Synth: Sustain	27	ADSR Sustain level
Synth: Release	28	ADSR Release time

Panorama Section

parameter	id	description
Pan String	29	Panorama position of *String* section
Pan Synth	30	Panorama position of *Synth* section
Pan Bass	31	Panorama position of *Synth Bass* section

Phaser Section

parameter	id	description
Phaser: Speed	32	Phaser speed
Phaser: Feedback	33	Phaser feedback
Phaser: String	34	Activates the Phaser for the *String* section
Phaser: Synth	35	Activates the Phaser for the *Synth* section
Phaser: Bass	36	Activates the Phaser for the *Synth Bass* section

Delay Section

parameter	id	description
Delay: Time	37	Delay time
Delay: Feedback	38	Delay feedback
Delay: Mix	39	Delay dry/wet mix
Delay: x2	40	Activates doubled delay time
Delay: Sync	41	Synchronizes delay time to host tempo
Delay: Vibe	42	Activates the delay vibe modulation
Delay: Ping-Pong	43	Activate the Ping-Pong mode

Tweak Sections

parameter	id	description
Chorus: Slow	44	Toggles between fast and slow Ensemble speed
Synth: ADSR to VCA	45	Activates the ADSR modulation of the *Synth* VCA
Chorus 1 Speed (Fast)	46	Modulation speed of the first delay line (fast)
Chorus 2 Speed (Fast)	47	Modulation speed of the second delay line (fast)

parameter	id	description
Chorus 3 Speed (Fast)	48	Modulation speed of the third delay line (fast)
Chorus 1 Depth (Fast)	49	Modulation depth of the first delay line (fast)
Chorus 2 Depth (Fast)	50	Modulation depth of the second delay line (fast)
Chorus 3 Depth (Fast)	51	Modulation depth of the third delay line (fast)
Chorus 1 Speed (Slow)	52	Modulation speed of the first delay line (slow)
Chorus 2 Speed (Slow)	53	Modulation speed of the second delay line (slow)
Chorus 3 Speed (Slow)	54	Modulation speed of the third delay line (slow)
Chorus 1 Depth (Slow)	55	Modulation depth of the first delay line (slow)
Chorus 2 Depth (Slow)	56	Modulation depth of the second delay line (slow)
Chorus 3 Depth (Slow)	57	Modulation depth of the third delay line (slow)
Delay Vibe Speed	58	Modulation speed of the delay’s Vibe effect
Delay Vibe Depth	59	Modulation depth of the delay’s Vibe effect

Frequently Asked Questions

How do I install Oxid *(Windows VST2 32 bit version)?*

Just copy the files `oxid.dll` from the ZIP archive you have downloaded to your system's or favorite DAW's VST2 plug-in folder. Your DAW should automatically register the *Oxid* VST2 plug-in the next time you start it.

How do I install Oxid *(Windows VST2 64 bit version)?*

Just copy the file `oxid64.dll` from the ZIP archive you have downloaded to your system's or favorite DAW's VST2 plug-in folder. Your DAW should automatically register the *Oxid* VST2 plug-in the next time you start it.

Note: You may have to remove any existing (32 bit) `oxid.dll` from your VST2 plug-in folder or else your DAW may screw the versions up…

How do I install Oxid *(Windows VST3 64 bit version)?*

Just copy the files `oxid.vst3` from the ZIP archive you have downloaded to your system's or favorite DAW's VST3 plug-in folder. Your DAW should automatically register the *Oxid* VST3 plug-in the next time you start it.

How do I install the Oxid *(Windows AAX 64 bit version)?*

Copy the file `oxid_AAX_installer.exe` from the ZIP archive you have downloaded to any of your system's folder and run it. Your AAX-enabled DAW (Pro Tools etc.) should automatically register the *Oxid* AAX plug-in the next time you start it.

How do I install Oxid *(Mac)?*

Locate the downloaded PKG package file in Finder (!) and do a right- or control-click on it. In the context menu, click on “Open”. You will be asked if you really want to install the package because it comes from an “unidentified developer” (me ☺). Click “OK” and follow the installation instructions.

What is the plug-in ID of Oxid*?*

The ID is **`oxid`**.

… the Release slider of the ADSR is not working?

As described in section *The Pitfalls*, the “RELEASE” parameter of the *String* section determines the maximum release time of the ADSR’s release time.