REndomPatch User Guide

Welcome to REndomPatch!

REndomPatch is a tool to create randomized patches for RackExtensions™ in Propellerhead's Reason™. In theory REndomPatch can create patches for every RE, as long as...

• The RE can load and save patches
• A RE definition file for REndomPatch is available

Please remember that only RackExtensions are supported. Reason's built-in devices (like Subtractor, Thor etc.) use another format and therefore are **not** supported. As a rule of thumb: everything that saves as “*.repatch“ can be used!

Requirements:

• MacOS X 10.7 or higher.
• Propellerhead Reason™ 6.5 or higher.
• Any of the following RackExtensions:

AdditiveOscillator
Antidote
Chip64
Combo X-705 Space Organ
Driver
Ivoks**
FM4
kHs ONE
Mixfood Orange
Mixfood Unison Xs
MonoPoly
Noxious
Oberon
Parsec
Polysix
Predator
PX7
Quad
Radical Keys
Radical Piano
Re-Tron
ReDominator
Revival
Snakebite*
TableOscillator
Tres
Uhbik-A
Uhbik-G
Vecto
VK-1 Viking
(more to come)

** There are actually two settings for Snakebite: 'Wide' and 'Narrow'. 'Wide' leads to mostly dissonant sounds, whereas 'Narrow' gives more musical results.*

*** Named as 'ИВОКС'. I prefer the russian style ;) .*

Credits

Many thanks to the following users for sharing their RE definitions:

kylelee (kHs ONE, Radical Keys, Radical Piano)

Version history

1.0
First release.

1.1.0
As version 1.0 randomized all parameters of a RE – leading into more or (mostly) less usable results – REndomPatch 1.1.0 now gives the user the possibility to carefully select *what* to randomize.

1.2.0
Improved layout possibilities in filter window (see Part 2, Chapter 2.1.1).
RE definitions adapted to new layout where suitable.
Minor changes.

1.2.1
Menu *Rack Extensions > Show in Finder* added. This opens the folder where RE definitions are saved. May be useful to find your selfmade definitions.

1.3.0
Menu *Rack Extensions* shows the last 10 used REs now. Also this menu is disabled now (as it should be) when a RE is opened.

Part 1 – Fun with REndomPatch

Getting started

REndomPatch consists of two parts:

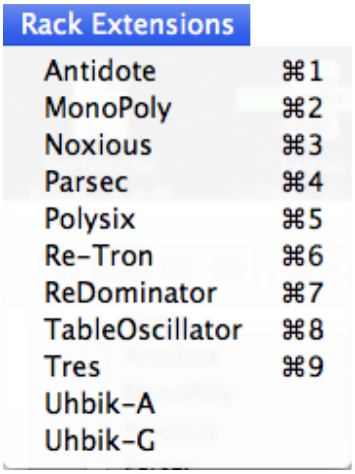
- the application itself
- the 'Definitions' folder

After downloading and unpacking REndomPatch, drag it to any place you like (most likely to the applications folder).

Start REndomPatch.

Select 'File > Import RE description...' or type Cmd+I and select any file in the 'Definitions' folder you need (one after another!!). These files describe the patch format a RE expects and will be copied into a system folder (/library/Application Support/REndomPatch to be more precise).

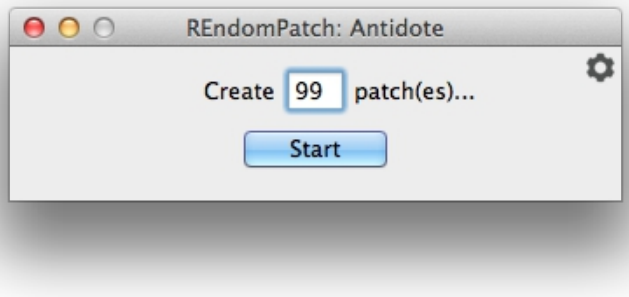
That's it for the moment. Depending on the RE descriptions you imported the RackExtensions menu now might look like this:



Before we proceed... some words about the Filter

As mentioned above, REndomPatch 1.0 originally randomised *every* parameter a RE uses. This was cool for a few of them, but for other REs the result was questionable: some REs provide just too much knobs to be useful when randomised!

REndomPatch 1.1.0 and above provides the ability to select what exactly to randomise – and this has to be taken literally! When you select a RE right after you imported it, and tell REndomPatch to create a number of patches...



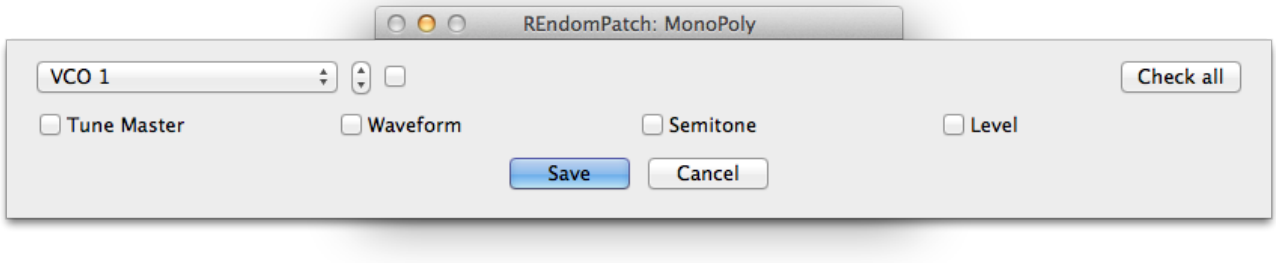
… you'll get a number of *Init Patch* sounds! That's just because all filters are switched off by default.

While coding this functionality I first did it the other way: everything was enabled by default and thus the patches have been... a little bit strange sometimes. And creating patches for more complex REs (e.g. Tres) was really a pain: many times I heared - *nothing*!

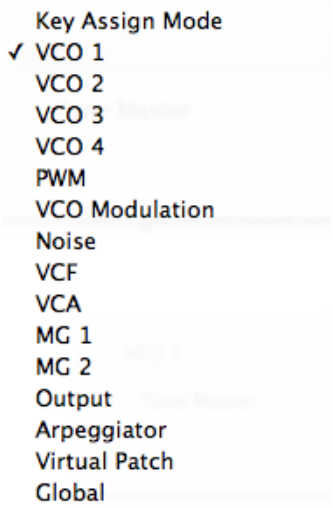
So, to prevent this frustration I introduced the clean way: starting with an *Init Patch* the user can proceed until the amount of chaos is reached which might be acceptable for him.

Using the Filter

To adjust the Filter to your needs click the little sprocket on the upper right corner of the above mentioned window or type Cmd+F *(I know that Cmd+F on Mac generally means "Find...", but it also fits for "Filter", so please look over this carelessness)*. You'll get a window like this...



The popup menu to the left showing "VCO 1" is used to select different parts of a RE (if available). Depending on what the guy did who made the RE Definition this may be an oscillator, a filter or a whole modulation matrix. My choice for MonoPoly looks like this...



The little up/down arrows next to it are used to select the previous/next module.

The checkboxes *beneath* show the knobs that are used in this module. You like the Waveform randomised? Just check it!!!

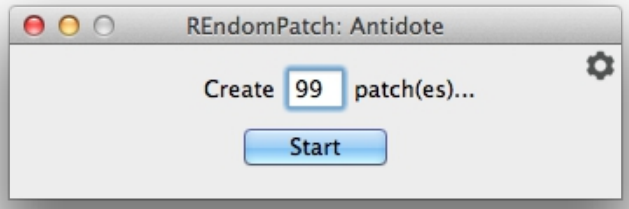
If you like to enable the whole module mark the checkbox next to the up/down arrows!

Now, what's this "Check all" button on the upper right? Well, that's for the hardcore guys: it checks all parameters in all modules – meaning it randomises ***everything*** like in REndomPatch 1.0! So be careful please.

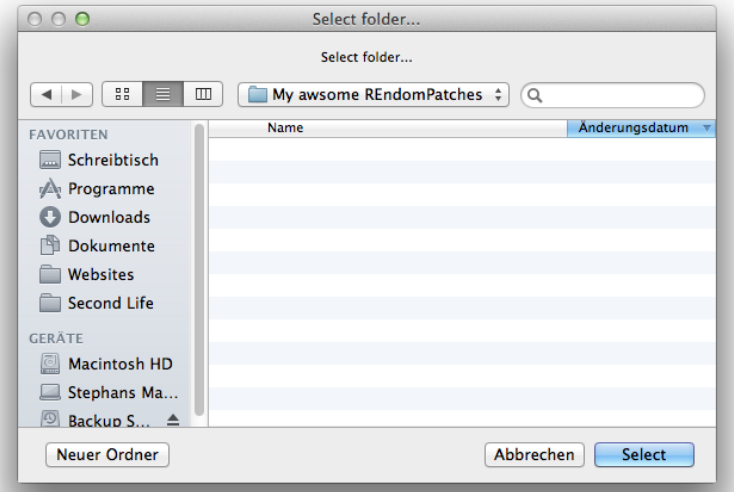
Hint: **If you use RE Definitions that are created for old REndomPatch 1.0, you have to hit the "Check all" button! If not your patches don't load in Reason!**

Creating random patches

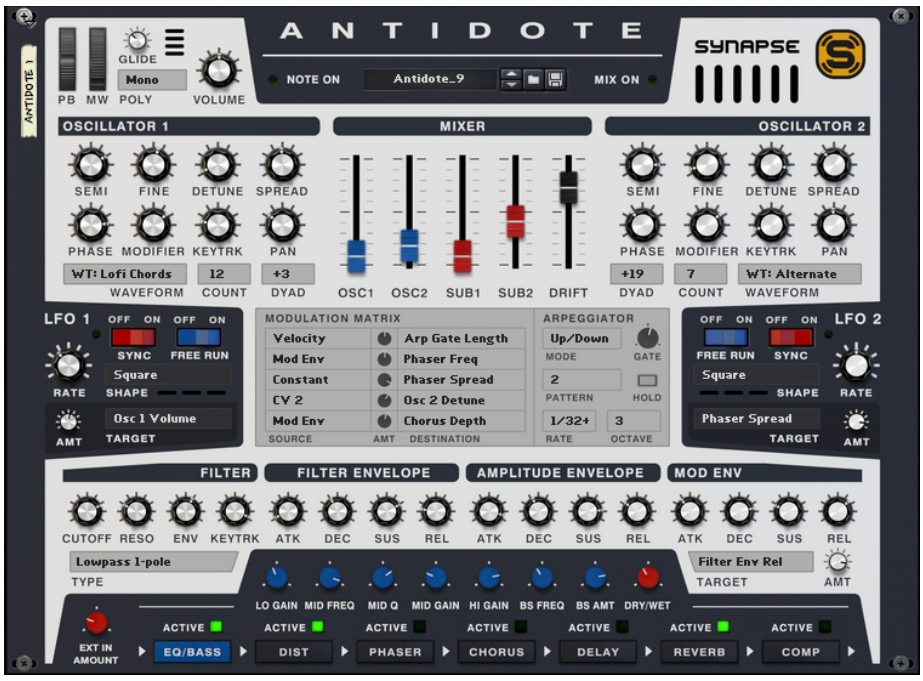
When you adjusted everything you suppose to need for your patches, save the settings. Type in the number of the patches you like to create...



… select the location where to save them and wait a second.



Locate the newly created sound(s) in the finder, doubleclick... and (if I did my job well) it should open in Reason. Yeah!



Misc stuff

REndomPatch is numbering the patches you create automatically – distinguishing between different REs. If you like to restart, select “File > Reset Numbers”.

Part 2 - How to create a RE definition for REndomPatch

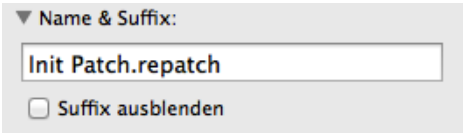
1.1 How Repatches work

In contrast to Reason's built-in devices (like Subtractor, Thor etc. …) which can't be used with REndomPatch, Repatches use a special text format called “XML”. This type of data can be opened and edited in a text editor (like TextEdit which is delivered with Mac OS). To show how a RE definition for REndomPatch is created I developed a fictional synth:

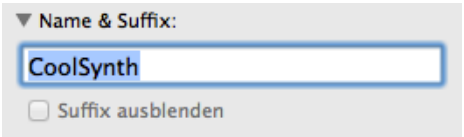
It's a very basic one consisting of one oscillator and a master section. Please forgive it's uglyness – I'm a musician, no visual artist! I promise to never design a real RackExtension ;-) !!!

Ok, we save the InitPatch to the desktop (just to find it easily). Before we procede let's rename it for future use in REndomPatch: select it, type Cmd+I and change it's name from InitPatch.repatch to the RE's name – without adding “.repatch” at the end!

Before



After



You will be asked if you really want to omit “.repatch”. Yes, confirm please!

Open it in TextEdit. Depending on how complex the RE is you work on, it will look like this:

```
<?xml version="1.0"?>
<JukeboxPatch version="1.0" >
  <DeviceNameInEnglish>
      CoolSynth
  </DeviceNameInEnglish>
  <Properties deviceProductID="com.JPsoft.CoolSynth"  deviceVersion="1.5" >
      <Object name="custom_properties" >
         <Value property="osc_wave"  type="number" >
             3
         </Value>
         <Value property="osc_tune"  type="number" >
             0.36789
         </Value>
         <Value property="volume"  type="number" >
             0.75
         </Value>
         <Value property="make_awesome"  type="boolean" >
             false
         </Value>
       </Object>
  </Properties>
</JukeboxPatch>
```

Oops, looks like rocket science??? Don't worry! What you see is a so-called XML-file as mentioned above. In short XML is a text format created to share data between applications or computers. HTML which is used to create websites in the internet is also some kind of XML, so you use it every day!

Let's have a closer look at it...

In line 3 to 5 we see the name of the RE:

```
<DeviceNameInEnglish>
    CoolSynth
</DeviceNameInEnglish>
```

This tells Reason how it should show up in the “Create...” menu.

The next line describes the manufacturer and the version of the RE

```
<Properties deviceProductID="com.JPsoft.CoolSynth"  deviceVersion="1.5" >
```

This helps Reason to keep all REs of one manufacturer together and place the dividing lines.

Parsec Spectral Synthesizer PX7 FM Synthesizer	Propellerhead
AutoArp	Black & Orange
VK-1 Viking Synthesizer	Blamsoft
Tres Mono Synth	FXpansion
Re-Tron	GeForce Software
Ammo 1200BR Modulation Synthesizer	Jiggery-Pokery
Korg MonoPoly Korg Polysix	Korg
Little LFO	Little IO Co.

All this is good to know but we don't touch it. More interesting to us is all that comes after the line...

```
<Object name="custom_properties" >
```

This section describes the values of all the knobs and switches we find in our RE. Let's have a closer look.

```
<Value property="osc_wave"  type="number" >
    3
</Value>
```

The first term `property="osc_wave"` is the name of a parameter the RE uses internally for it's calculations. Most developers use names that are pretty clear. Looking at CoolSynth's front plate we find that this corresponds to the oscillator's waveform switch.

The next term `type="number"` describes the *kind* of a value. In Repatches there are two different ones: numbers and boolean. Numbers seem to be clear if we look further: it may be 1, 2, 3... or something like 0.36789, 0.75 etc.

But what the heck is “boolean”?? Well, that's a kind of mathematical expression invented by british mathematician George Boole (1815-1864). It describes decisions which may be *true* or *false*, or switches that can be *On* or *Off*. As computers internally work with the numbers *0* and *1* only, boolean expressions are a “bread and butter”-technology in computer science.

The next line after `<Value property="osc_wave"  type="number" >` shows the value a certain parameter uses in the patch. For instance 0.75 means that a knob is turned up by 75%.

The last line

```
</Value>
```

marks the end of the description of this single parameter.

1.2 The little difference – numbers in REs and numbers in REndomPatch

As we've seen in chapter 1.1 a RE has two possibilities for “type”: numbers and boolean. Numbers can be 1, 2, 3... or 0.36789, 0.75 etc. The big difference between them is that some of them use a point and some of them don't. Those *without* a point are called “integer”, those *with* a point are “floating point numbers” or just “float”. As we've seen in the XML-file REs don't make a difference here (there are only “numbers”) – just because the RE knows in advance what kind of number to expect. But to tell REndomPatch what kind of number we need at a certain place we'll have to be a little bit more precise.

1.3 Why using what

So let's see what types and numbers are used in our patch...

```
<Value property="osc_wave"  type="number" >
    3
</Value>
<Value property="osc_tune"  type="number" >
    0.36789
</Value>
<Value property="volume"  type="number" >
    0.75
</Value>
<Value property="make_awesome"  type="boolean" >
    false
</Value>
```

`osc_wave` describes the switch on the front panel to select the oscillator waveform. The switch has five possible states to select either sine, triangle, pulse, saw or noise.

This ideally can be described by *integer* numbers. As there are five possibilities you might think that it's just 1, 2, 3, 4, 5 but **it is not**! Numbers in REs (no matter if integer or float) always start with '0', so the list for oscillator waveform looks like this:

0 = Sine
1= Triangle
2 = Pulse
3 = Saw
4 = Noise

Please keep this in mind!

`osc_tune` is a knob that lets you smoothly tune from -1 octave to +1 octave. This can be described best by a float number – because it's a *smooth* movement! Float numbers in REs always range from 0 to 1 with all values in between. In this case it looks like this:

0 = -1 Octave
.
.
.
0.5 = in tune
.
.
.
1 = +1 Octave

On the right side of CoolSynth's front panel in the master section we find the “Make awesome” switch. In the XML file it's described by `make_awesome`. Because the switch can be *On* or *Off*, it's marked as boolean:

false = Off
true = On

Please be aware that On/Off states *can* be described by boolean expressions, but they can also be described by integer numbers ranging beeing 0 or 1:

0 = Off
1 = On

Both methods do the same and it's up to the developers which one they use. So always be careful what type is expected!

The master volume finally is also a knob made for smooth operation and is represented by a float number ranging from 0 to 1 (0 = no sound, 1 = maximum level).

Note: You may have noticed that the order of the parameters in the XML file is different to how they appear on the front panel. I'll discuss this later.

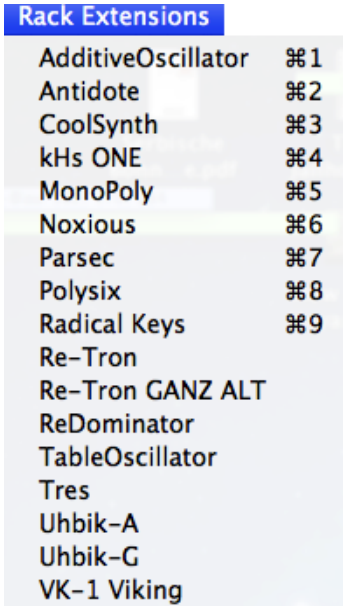
2.1 How to tell REndomPatch what to do

After this little journey into computer science and history, it's time to use our knowledge for REndomPatch!

First of all we've to do a little preparation again:

1. Open the XML file in TextEdit if not already done.
2. Start REndomPatch.
3. Go to “File > Import RE description...” and import the XML file. The file will be moved to REndomPatch's internal folder (/library/Application Support/REndomPatch), but still remains open in TextEdit. That's handy because you can edit and test it at the same time.

CoolSynth should be available in the RackExtensions menu now:



<-------- Here it is!!

Note: As CoolSynth doesn't exist as RE in reality and therefore can't be used in Reason you may prefer an existing one instead. For convenience I'll proceed with CoolSynth for now.

Let's look at the XML code again:

```
<?xml version="1.0"?>
<JukeboxPatch version="1.0" >
  <DeviceNameInEnglish>
      CoolSynth
  </DeviceNameInEnglish>
  <Properties deviceProductID="com.JPsoft.CoolSynth"  deviceVersion="1.5" >
      <Object name="custom_properties" >
         <Value property="osc_wave"  type="number" >
             3
         </Value>
         <Value property="osc_tune"  type="number" >
             0.36789
         </Value>
         <Value property="volume"  type="number" >
             0.75
         </Value>
         <Value property="make_awesome"  type="boolean" >
             false
         </Value>
       </Object>
  </Properties>
</JukeboxPatch>
```

To tell REndomPatch what to do when creating patches, we need to discribe what type to create (number beeing integer or float, or boolean). We also need to describe the range of the numbers the RE expects. And of course we need to tell REndomPatch how each parameter is used in the filter section!

This can be done with an expression like this:

```
{{integer|0|4|Waveform|Oscillator|3}}
```

Damned, what's this? Don't worry, it's pretty easy!

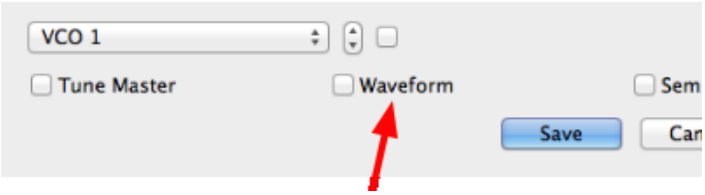
As you see we deal with the oscillator waveform switch. As seen in chapter 1.3 this offers the following possibilities:

0 = Sine
1= Triangle
2 = Pulse
3 = Saw
4 = Noise

The first part `integer` shows what *kind* of number to use.

The numbers 0 and 4 describe the minimum and maximum values this number can be.

“Waveform” is the label that is shown in the filter window.



(example from Part 1)

“Oscillator” means to which module in the filter window the switch belongs to.

(also from Part 1)

The last part – in this example '3' – tells REndomPatch the default value for Waveform, meaning what value to use when the filter for this is *not* selected. 3 stands for sawtooth, so of you decide to not randomise the waveform switch you'll always get sawtooth.

In order to tell REndomPatch what to do with a certain parameter we just have to replace it's value in the XML code by the above mentioned term:

```
<Value property="osc_wave"  type="number" >
    {{integer|0|4|Waveform|Oscillator|3}}
</Value>
```

The next parameter – this time it's a float number – will look like this:

```
<Value property="osc_tune"  type="number" >
    {{float|0|1|Tune|Oscillator|0.5}}
</Value>
```

Again, the first part says what type of value we need: float. As mentioned before float in Repatches always range from 0 to 1. It will be labled as *"Tune"* in the *"Oscillator"* section, and if not selected it will be *0.5* = not detuned.

Next in the XML code is the volume knob. It looks like this now:

```
<Value property="volume"  type="number" >
    {{float|0|1|Volume|Master|0.75}}
</Value>
```

As mentioned above it's also a floating point number from 0 to 1. It's label is *"Volume"*, but this time it belongs to the *"Master"* section. *0.75* means that it's turned 75% up.

Finally the XML code shows the description for the "Make awesome" switch. Remember that this one is of boolean type. The code for REndomPatch looks this way:

```
<Value property="make_awesome"  type="boolean" >
    {{boolean|0|1|Make awesome|Master|0}}
</Value>
```

Oops, didn't I say that boolean can be either *true* or *false*? Obviously it should, because the original code was

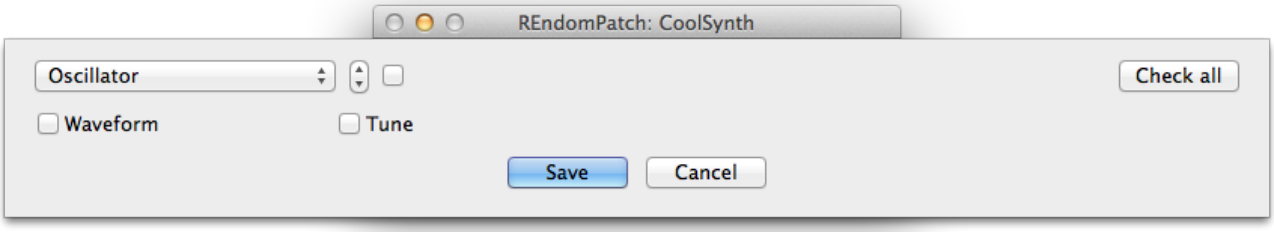
```
<Value property="make_awesome"  type="boolean" >
    false
</Value>
```

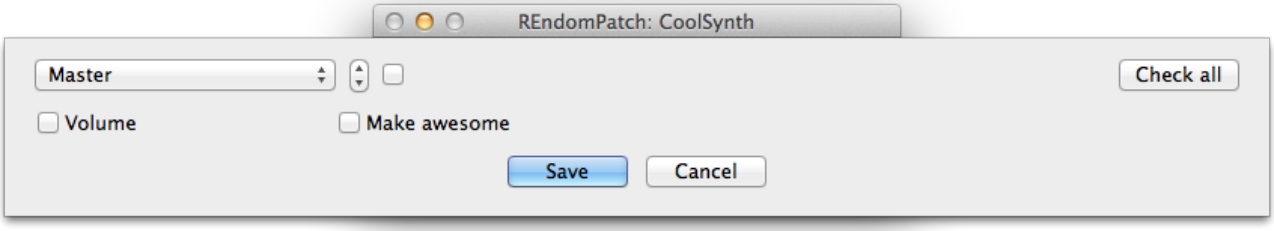
Why the heck do we need 0 and 1 here instead?? Well, it was just a little carelessness of my part when I developed REndomPatch! So, you have to use 0 and 1 when defining boolean values, but REndomPatch will convert them correctly into *true* or *false*. Sorry for this inconvenience!!

After all is done our code looks like this now:

```
<?xml version="1.0"?>
<JukeboxPatch version="1.0" >
  <DeviceNameInEnglish>
      CoolSynth
  </DeviceNameInEnglish>
  <Properties deviceProductID="com.JPsoft.CoolSynth"  deviceVersion="1.5" >
      <Object name="custom_properties" >
         <Value property="osc_wave"  type="number" >
             {{integer|0|4|Waveform|Oscillator|3}}
         </Value>
         <Value property="osc_tune"  type="number" >
             {{float|0|1|Tune|Oscillator|0.5}}
         </Value>
         <Value property="volume"  type="number" >
             {{float|0|1|Volume|Master|0.75}}
         </Value>
         <Value property="make_awesome"  type="boolean" >
             {{boolean|0|1|Make awesome|Master|0}}
         </Value>
       </Object>
  </Properties>
</JukeboxPatch>
```

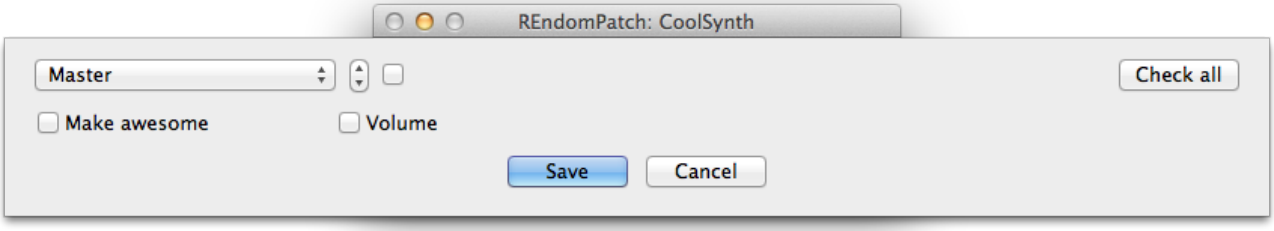
Go to REndomPatch and have a look at the filter window!





As said before, the “Make awesome” switch on CoolSynth's front panel is located left to the Volume knob, but here it's vice-versa! That's because in the XML file the volume parameter stands *before* `make_awesome`. Sometimes developers are a little bit inaccurate, so that the parameters may be “somewhere” in the XML file. But that's no problem because The RE is clever enough and doesn't care how things are sorted. So just exchange them and it will look like this:

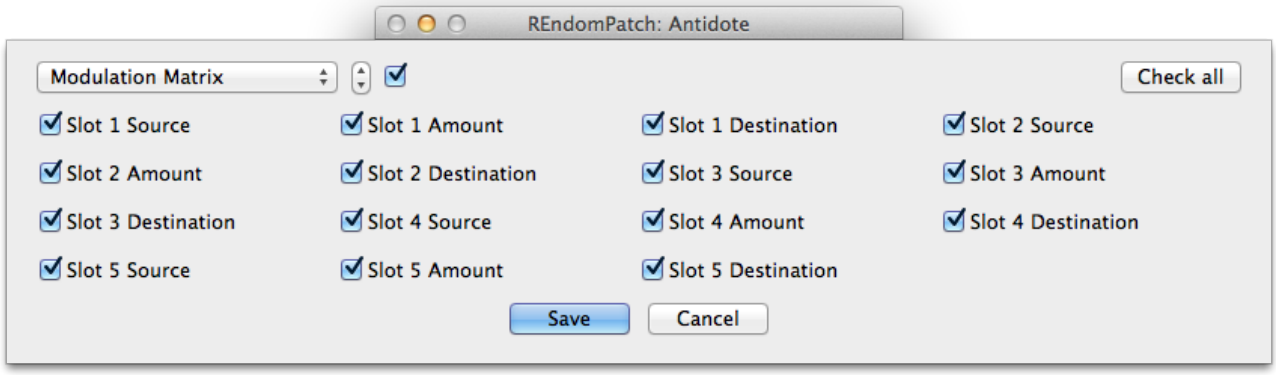
```
<?xml version="1.0"?>
<JukeboxPatch version="1.0" >
  <DeviceNameInEnglish>
      CoolSynth
  </DeviceNameInEnglish>
  <Properties deviceProductID="com.JPsoft.CoolSynth"  deviceVersion="1.5" >
      <Object name="custom_properties" >
         <Value property="osc_wave"  type="number" >
             {{integer|0|4|Waveform|Oscillator|3}}
         </Value>
         <Value property="osc_tune"  type="number" >
             {{float|0|1|Tune|Oscillator|0.5}}
         </Value>
         <Value property="make_awesome"  type="boolean" >
             {{boolean|0|1|Make awesome|Master|0}}
         </Value>
         <Value property="volume"  type="number" >
             {{float|0|1|Volume|Master|0.75}}
         </Value>
       </Object>
  </Properties>
</JukeboxPatch>
```



Yeah, we got it!!

2.1.1

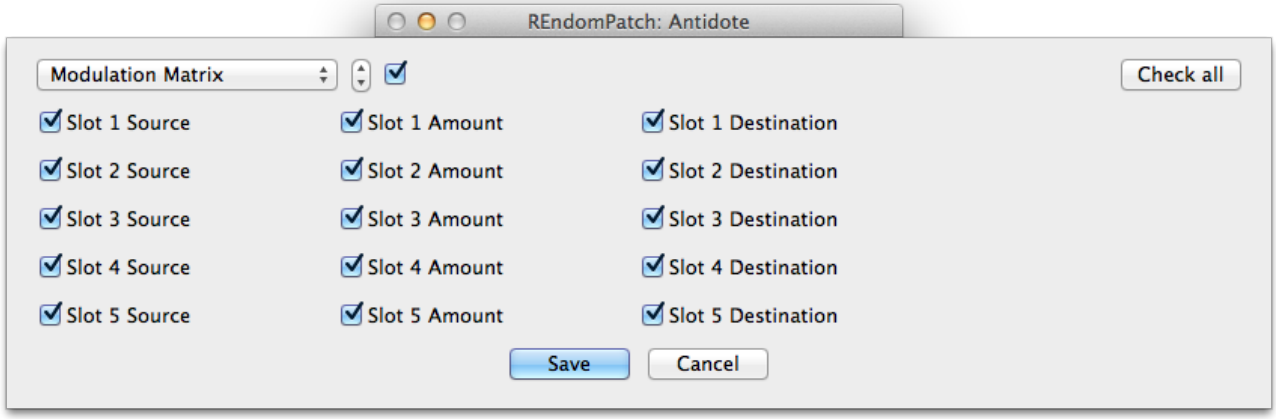
In REndomPatch 1.2.0 there was introduced an addition *optional* parameter:
Let's say we got a filter window looking like this:



Antidote's Modulation Matrix provides five slots where every one consists of three parameters (Source, Amount, Destination). As REndomPatch adds a new line after every 4th checkbox automatically this might look a bit odd. So there's the new `|nl` command now:

```
{{integer|0|1|Slot 2 Amount|Modulation Matrix|0|nl}}
```

This tells REndomPatch to create a new line for `"Slot 2 Amount"`. If this is done for every `"Slot n Amount"`, the window looks like this:



As already mentioned this command is **optional**. Just use it when needed. And beeing optional all previous made RE definitions stay compatible – they just look the old way.

Hint: Sometimes you my need more than one new line at a time. This can be done by using multiple nl's:

```
{{integer|0|1|Slot 2 Amount|Modulation Matrix|0|nl}} adds one new line
{{integer|0|1|Slot 2 Amount|Modulation Matrix|0|nlnl}} adds two new lines
{{integer|0|1|Slot 2 Amount|Modulation Matrix|0|nlnlnl}} adds three new lines etc.
```

2.2 Finding the proper numbers

Sometimes it may be difficult to find the correct number types. For instance if you see '1' in a Repatch you can't say for sure if it's integer or float! Even if "1" represents a float, it's displayed as "1" and not as "1.0" as one might think. So to be sure to use the correct type, locate the specific knob in Reason, turn it up a little bit (somewhere between all down and all up) and save the patch. Open it in TextEdit, find the parameter and look what it is! If it contains a point (e.g. 0.36587), it's float. Otherwise it's integer. BTW numbers *larger* than 1 are *always* integer!

The same procedure can be used to determine the range of integer numbers. CoolSynth offers five waveforms. You can easily count them. But what if another synth provides 99 waveforms instead? No problem: select the last waveform in Reason, save the patch and look at the XML code!

2.3 Limiting parameter ranges

Sometimes you'll find that randomising may 'cause pretty odd patches. For instance there are synths that provide attack times of up to 30 seconds. So if you use the full range (0 to 1) you'll get sounds that start very slowly most of the time. In this situation it might be a good idea to limit the maximum possible value to get more useful results. Go to Reason and adjust the Attack (or whatever) knob to the time you like – let's say 5 seconds. Save the patch, open it in TextEdit and look at the value it uses now. Go to your XML file and change the maximum range accordingly, like this:

```
{{float|0|0.35|Attack|Envelope|0}}
```

BTW you can see in this example, that the numbers a RE uses internally and the number that is displayed on the front panel not necessarily correspond to each other. If it's a volume knob from 0%... 50%... 100% this may be represented by 0... 0.5... 1, but in many cases it's different. A good example is the Antidote synth: you have to turn the Attack knob about 75% up to reach 1 second, the remaining 25% cover 1 – 10 seconds!

2.4 Exclude parameters

In our CoolSynth model we prepared the volume knob for randomisation. But in reality, meaning the RE descriptions I made to deliver with REndomPatch, I didn't do that, because I think it's more convenient to leave this thing untouched: you always want to *hear* the patches, that's what they are made for! To do this, leave the original settings just how they are:

```
<Value property="volume"  type="number" >
    0.75
</Value>
```

Note: This has changed since the filter was introduced, meaning that Volume IS available now. But the fact remains the same: you still can leave parameters untouched. This is useful 'cause sometimes developers leave dummy parameters in the Repatch that are needed for compatibility but don't do anything.

2.5 Finally: testing, testing, testing!

REs are extremely chicken-hearted concerning all those numbers. If integer is expected and it gets float instead, the patch doesn't load. If it expects an integer ranging from 0 to 5 and gets 6 instead, the patch doesn't load. If it needs true or false and gets 0 or 1 the patch doesn't load. This might become a pain if you notice it too late: where the heck did I fail when working on the XML file?

That's why we opened the file in TextEdit first and then imported it into REndomPatch at the beginning! Now we can create some randomised patches after every line we edited with no need to re-import – and I highly recommend this! Let REndomPatch create a sound after every change you make and check it with filter on and filter off. When all is done, create a larger number of patches and look if they all load in Reason one after another.

3.1 Ask & Share!

I hope you understood everything. If not, feel free to ask me whatever you need to know, and tell me all the mistakes I made in this description and how I can do better. You'll find me in the Propellerheads User Forum https://www.propellerheads.se/forum/member.php?u=57247 .

Also be envited to *share* the RE descriptions you create! Depending on the complexity of a RE you'll find that it may be a lot of work to do this (e.g. I worked half a night on TRES!), so other users may be happy to avoid this pain.

That's all for the moment.

Kind regards,
Joshua Philgarlic.