Mod Documentation
SashaT1804 - CustomChannelInjector_v3.1

Custom Radio & TV Channels Injector (& Video Games too)

Doc: v3.1.1; Mod: v3.1
May 3, 2022

Mod Presentation

The **Custom Channel Injector** mod aims to facilitate the addition of custom radio and TV channels to *The Sims™4*. It is mostly made for fellow modders, as it still requires the creation of the XML tuning files for both the channels and the corresponding listening interactions. It also requires an extra snippet tuning file, that is explained in details in this documentation. If you are not familiar with XML modding yet, I recommend first experimenting, with Sims 4 Studio for example, and making sure your channels work fine on custom objects. What this mod provides is, once you are done with the XML files, it will **automatically** add your custom radio channels to **every stereo object** in game, and add your custom TV channels to **every TV object**. I have also added video games, on request from someone, as it is the same kind of injection. It is injecting with a script, **not overriding** any file, so it is compatible with any mod already touching to channels or stereo, TV, and computer objects. It will also add to all related custom items, no matter where they come from, as long as they are done properly. I am releasing this mod for free, but if you want to use it for your own released mods, please do not directly include the script file, but link to my download page. This will give me some much needed visibility as I am really just starting, and it will also prevent users from having several copies of the script in several mods, which would lead to problems. This is my very first mod, I hope it will be as useful for other modders as it is for me.

-Sasha

Contents

1 Update Notes

v3.1 - May 3, 2022

Functionalities

- Added an option to make channels non dance-able.

Bug corrections

- Corrected an issue with DJ booths, the music would play but the object would officially stay in OFF state and the music animation thing was not showing up.

v3.0 - April 22, 2022

Functionalities

- Added support for **computer** video games.

Bug corrections

- The instance ID of the string table in the test package was set to 0, which might have caused some issues if some other mod had done the same. I replaced it with a proper hash.

v2.0 - April 16, 2022

Functionalities

- Added support for TV channels.

v1.1 - April 11, 2022

Functionalities

- Radio channels can now be added to the earbuds, hot tubs, smart hub, water scooters, skating rings, and DJ booths.

- In theory, radio channels can be added to bots and humanoid robots as well. My best guess is that bots are the small robots we can craft, and the humanoid robot would be the Servo, but it takes too long to create one so I did not test it yet. If you do test it, please let me know whether it works or not.

Bug corrections

- Solved an issue that prevented the radio channels from showing up on the 2 entertainment centers from *Tiny Living*.

2 Prerequisites

I will not go through the details of general XML modding, so if you are not familiar with this, it is better to first experiment with custom objects before switching to this mod.

2.1 Channel Object State Tuning

You will first need to create the channel XML file. This is done with an *Object State Tuning*, with class *AudioChannel* for radio channels, and *VideoChannel* for TV channels. For example, here is the file corresponding to the in-game Retro channel:

```
<?xml version="1.0" encoding="utf-8"?>
<I c="AudioChannel" i="object_state" m="objects.components.state" n="StereoChannel_Retro" s="32571">
  <V n="_display_data" t="enabled">
    <U n="enabled">
      <V n="instance_display_description" t="enabled">
        <T n="enabled">0xF32A0A2A<!--Tunes from the ol' days--></T>
      </V>
      <V n="instance_display_icon" t="disabled" />
      <V n="instance_display_name" t="enabled">
        <T n="enabled">0x4CFEBAB4<!--Retro--></T>
      </V>
    </U>
  </V>
  <L n="buff_weight_multipliers">
    ...
  </L>
  <U n="new_client_state">
    <V n="audio_state" t="apply_new_value">
      <V n="apply_new_value" t="start_audio">
        <U n="start_audio">
          <T n="audio" p="InGame\Audio\Music\stereo\simretro\mu_stereo_simretro.propx">
            39b2aa4a:00000000:435fa71eee9c0b62
          </T>
        </U>
      </V>
    </V>
    <V n="broadcaster" t="apply_new_value">
      <V n="apply_new_value" t="start_broadcaster">
        <U n="start_broadcaster">
          <L n="broadcaster_types">
            <U>
              <T n="item">74334<!--broadcaster_Reaction_LoudNoises_TVStereo--></T>
            </U>
            <U>
              <T n="item">258533<!--broadcaster_Stereo_Retro--></T>
            </U>
          </L>
        </U>
      </V>
    </V>
  </U>
  <L n="super_affordances">
    <T>194659<!--autoPicker_MusicProductionStation_PlayTrack--></T>
  </L>
</I>
```

You will need to create one for your channel, and edit the description, name, audio/video file, etc. I will not go through this, as many tutorials already exist. Once you have the file, note the tuning id ("s" attribute, for the Retro channel it was 32571). This id will be needed later in the snippet tuning.

2.2 Interaction Tuning

Then, you will need to create the listening/watching interaction corresponding to your new channel. This is done with an *Interaction Tuning*, with class *ListenSuperInteraction* for radio channels, and *WatchSuperInteraction* for TV channels.

For example, here is the header of the interaction to listen to the Retro channel:

```
<?xml version="1.0" encoding="utf-8"?>
<I c="ListenSuperInteraction" i="interaction" m="objects.electronics.stereo" n="stereo_listen_retro" s="32572">
```

You will need to create one for your channel, edit whatever you want to change, and link the previously created object state in the "required_station" field. Also note the tuning id for the interaction, as it will also be needed later.

3 Snippet Tuning

3.1 General Format

The extra snippet file is pretty simple:

```
<?xml version="1.0" encoding="utf-8"?>
<I c="CustomChannelsInjector" i="snippet" m="channelinjector" n="[name]" s="[tuning id]">
  <L n="radio_channels">
    ...
  </L>
  <L n="earbuds_channels">
    ...
  </L>
  <L n="hot_tub_channels">
    ...
  </L>
  ...
</I>
```

Just make sure you keep the "c" and "i" attributes exactly as they are. You can set the name attribute as you wish, and take the hash as usual for the tuning id. To add radio channels, you then need to fill the "radio_channels" part.

3.2 Radio Channels

3.2.1 Main usage

For each of your custom radio channels, you should have one channel id, and one interaction id. Simply add all of them as follows:

```
<L n="radio_channels">
  <L n="channel_list">
    <T>[channel id 1]</T>
    <T>[channel id 2]</T>
    ...
  </L>
  <L n="interaction_list">
    <T>[interaction id 1]</T>
    <T>[interaction id 2]</T>
    ...
  </L>
</L>
```

You can have as many as you want. Just put the channel ids in the channel list, and the interaction ids in the interaction list. This will add your channels to every *standard* stereo and *wall* stereo item in game, including the entertainment centers from *Tiny Living* (the bookshelf/TV/stereo/whatever combos).

3.2.2 Non-standard stereo objects

v1.1 introduces new options to add channels to other items that can play radio channels. The options are the following:

- **radio_channels:** Channels to add to normal and wall stereo objects;
- **earbuds_channels:** Channels to add to earbuds;
- **smart_hub_channels:** Channels to add to the smart hub object, that's the stereo object called "Lin-Z Smart Speaker" in game, the one you can be friends with;
- **water_scooter_channels:** Channels to add to water scooters (yeah, those can play music, I didn't even know);
- **hot_tub_channels:** Channels to add to hot tubs;
- **skating_ring_channels:** Channels to add to skating rings;
- **dj_channels:** Channels to add to DJ booths;
- **bot_channels:** Channels to add to bots (the small robots that can be crafted with the robotics skill);
- **humanoid_channels:** Channels to add to humanoid robots (the Servo, I think, but I didn't check)

It all works the same, just give the list of channels and the list of interactions for each category you want. Each category requires a **different channel** definition and a **different interaction**, so if you have one channel that you want on all the objects, you still need to define several interaction tunings and several object state tunings. That is how it works inside, there is nothing I can do about it. The original channels also have that many files, unfortunately, because each object type requires some tweaks (usage of remote or not, or necessity to be close to the object to turn it on, for example).

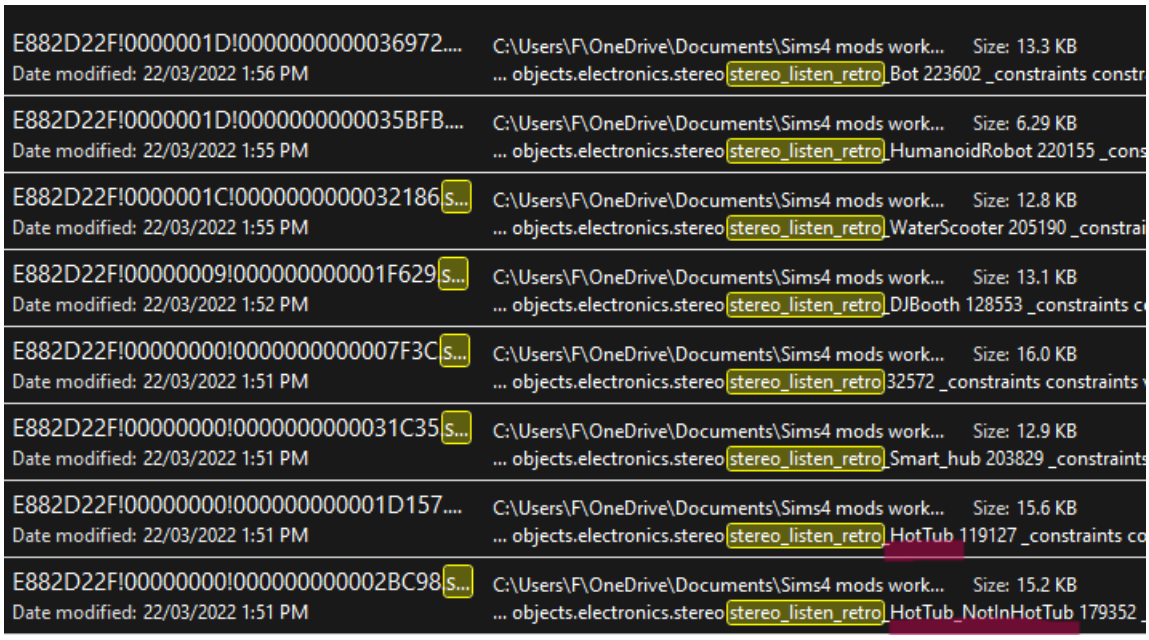
The order of the options do not matter at all, you can first have the earbuds channels and then radio ones, etc. The order of the channel ids and interactions ids do not need to match either, this would work the same:

```
<L n="radio_channels">
  <L n="channel_list">
    <T>[channel id 1]</T>
    <T>[channel id 2]</T>
    ...
  </L>
  <L n="interaction_list">
    <T>[interaction id 2]</T>
    <T>[interaction id 1]</T>
```

```
    ...
  </L>
</L>
```

Just so you can understand better, below is the list of original files for the *Retro* channel. You can start by duplicating them all and adjusting them to your channel.

In addition, the *hot tubs* require 2 interactions, one for playing from inside the tub, and another one for playing from outside the tub. This is the case for every channel as well. See below the original listen interactions related to the *Retro* channel.



Note that all the options are, well, *options*. So none of them are necessary. If you are satisfied with your channels only showing up on normal stereo objects, then you only need the "radio_channels" one, and no need to bother with the rest. It is also totally possible to add channels specifically for hot tubs, or DJ booths, or any category of objects. There is no need to create them all if you do not have the use for it.

3.2.3 Making channels non dance-able

v3.1 introduces the possibility to make stereo channels non dance-able. Basically, that deactivates the *"Dance"* and *"Dance the Rumbasim"* interactions while these channels are playing. Some radio channels from the original game are non dance-able, like the Classical and Lullaby channels for example. This makes sense mostly for calm music, as the energetic dancing really does not fit and it just looks weird. It was bothering me, so I added this option.

To make channels non dance-able, simply use the new **make_not_danceable** option, as follows:

```
<L n="make_not_danceable">
  <L n="channel_list">
    <T>[channel id 1]</T>
    <T>[channel id 2]</T>
    ...
  </L>
</L>
```

Only the channels are needed, I then inject them to the blacklist of the dance interaction.

The dance interactions seems to only show up on normal stereo objects, the smart hub, and DJ booths. I have **not tested this on DJ booths**, because it does not make sense to have non dance-able channels on them in the first place. But adding the normal and the smart-hub channel ids to this option will deactivate the dancing on these 2 types of objects for the given channels.

3.3 TV Channels

The TV channels work exactly the same way, there is simply a new option called **tv_channels**. So, just do the following:

```
<L n="tv_channels">
  <L n="channel_list">
    <T>[channel id 1]</T>
    <T>[channel id 2]</T>
    ...
  </L>
  <L n="interaction_list">
    <T>[interaction id 1]</T>
    <T>[interaction id 2]</T>
    ...
  </L>
</L>
```

The order of the options do not matter at all, you can first have the TV channels and then radio ones, etc. The order of the channel ids and interactions ids do not need to match either, this would work the same:

```
<L n="tv_channels">
  <L n="channel_list">
    <T>[channel id 1]</T>
    <T>[channel id 2]</T>
    ...
  </L>
  <L n="interaction_list">
    <T>[interaction id 2]</T>
    <T>[interaction id 1]</T>
    ...
  </L>
</L>
```

It seems there is only one type of objects that can display TV channels, namely, TVs, so only one channel and one interaction files are needed. If you find some objects that can display normal TV channels but not the custom ones, do let me know which ones so I can have a look.

3.4 Video Games

3.4.1 Computer Games

The overall usage for video games is similar, but there are a few extra attributes. See below:

```
<L n="computer_games">
  <L n="channel_list">
    <T>[channel id 1]</T>
    <T>[channel id 2]</T>
    ...
  </L>
  <L n="interaction_list">
    <T>[interaction id 1]</T>
    <T>[interaction id 2]</T>
    ...
  </L>
  <L n="novice_interaction_list">
    <T>[interaction id 3]</T>
    <T>[interaction id 4]</T>
    ...
  </L>
  <L n="pro_interaction_list">
    <T>[interaction id 5]</T>
    <T>[interaction id 6]</T>
    ...
  </L>
  <L n="test_set_list">
    <T>[test set id 1]</T>
    <T>[test set id 2]</T>
    ...
  </L>
</L>
```

Here is what the attributes are for:

- **channel_list:** As usual, the list of channels to be added, as video games are basically video channels that play on the screen while playing;
- **interaction_list:** The list of interactions to add without leveling conditions (typically: Play, Play GamedOut, Mod, Livestream, and Stream Let's Play, for each game)

- **novice_interaction_list:** List of *Novice tournament* interactions, have to be separated because the novice tournaments are unlocked by leveling the Video Gaming skill;

- **pro_interaction_list:** List of *Pro tournament* interactions, have to be separated because the pro tournaments are unlocked by leveling the Video Gaming skill, and later than the novice ones;

- **test_set_list:** List of the *Game Modded* TestSetInstance tunings (only the GameModded ones, not the normal ones). These are supposed to be added to objects, so I need them. You should have one for every VideoChannel Object State Tuning in the channel_list, and they **MUST BE IN THE PROPER ORDER**, so the first channel of the list will be associated with the first test set of the list, etc. I know this is not convenient because error-prone, I will try to solve it soon.

And just so you guys can better understand the many interactions that go with video games, here is what they are for:

- **Play:** The basic playing interaction;

- **Play GamedOut (Force self to play):** Replaces the basic playing interaction when the Sim has already been playing too much before;

- **Mod:** In the **Program** category on the computer, allows to mod the game;

- **Livestream:** Special interaction available for Sims in the Tech Guru career (not sure if there is a career level requirement to unlock, and it seems to appear in both branches);

- **Stream Let's Play:** Special interaction for Sims in the Social Media career, Internet branch, unlocked at level 8 of the career;

- **Novice tournament:** Just like the name says, handles the novice tournaments for that game;

- **Pro tournament:** Same for Pro tournaments.

Note about the interactions categories for the extra snippet: Putting the Tournament interactions in the main interactions list will totally work, but then these interactions will be available regardless of the Video Gaming skill level. Similarly, putting any interaction in the Novice or Pro interactions list will work, and will cause this interaction to be unlocked at level 1 or 7 respectively (I need to check the precise levels, it's something like that, but it might be 6 instead of 7 etc, I'll update the doc if the numbers are wrong).

3.4.2 Console Games

Coming soon

4 Installation

4.1 Script

Take the **SashaT1804_CustomChannelInjector_v3.1.ts4script** file, and simply add it to your mod folder.

4.2 Tunings

Create a .package file with all your XML tunings. Don't forget to add the extra CustomChannelsInjector snippet with them. Then, add this .package file to your mod folder.

It is an in-game injection, **not an override**, so it is compatible with any mod overriding channel lists or related object tunings.

5 Test package

I provided a test package, along with this mod and the documentation. To check that the injector is working properly, simply add the **ChannelsInjectorTest.package** file to your mod folder. This should add a new radio channel called *"Channel Injector Test"* on all in-game stereo objects, as well as a *"Channel Injector [category] Test"* channel on every other object that can play the radio. The smart hub channel is set as non danceable. It will also add a TV channel called *"TV Channel Injector Test"* to TV objects. With version 3.0, a new video game called *"Game Injector test"* has been added as well, along with all the related interactions (Play game, Mod game, Force self to play game, Novice game tournament, Pro game tournament, Livestream game, and Stream Let's Play game). You can use this test package as a base for your own custom channel, or for troubleshooting if needed.

6 Acknowledgements

I used the code from Scumbumbo's XMLInjector as a learning tool to create this mod. I'm all new to the modding community, but it seems he was very involved and appreciated, so a big thanks to him even though I will never get the chance to interact with him.

Also thanks to June Hanabi for the scripting tutorials and the project template that were really super helpful.