Fallout 4 Creation Kit: How to Create a Custom Radio Station

By kaotick_jay

Original Steam guide



This guide covers everything you need to create a custom radio station in the Fallout 4 Creation Kit, from setting it up in the editor to getting it working in-game on custom radios, jukeboxes, and the Pip-Boy.

It also includes my enhanced random shuffle script, adapted from the framework by YouTuber HeartImpaled. While the original framework provides a solid foundation, I've significantly improved the randomization and replay-avoidance logic to create a more natural listening experience with fewer repeated tracks.

There are plenty of tutorials that explain individual pieces of the process, but I found that none of them covered everything needed to build a fully functional radio station from start to finish. After a great deal of trial and error, countless hours watching tutorials, and extensive research through the official Creation Kit documentation, I finally put together a complete workflow.

The goal of this guide is to save you that time by providing a single, comprehensive reference that takes you from an empty project to a working radio station with proper track shuffling and support for both music and DJ or advertisement segments.

Updated script edition

The guide text and all images are reproduced from Steam. Sections 10-12 have been updated to use the current local PlagueRadioShuffler.psc and its matching setup instructions. Original screenshots are retained. The unchanged source compiled successfully with the official Fallout 4 Papyrus compiler 2.8.0.4: 1 succeeded, 0 failed. The separately generated PEX was not installed or tested in-game; scene-fragment compatibility remains unverified.

1) Intro and Quest Creation



This guide will walk you through creating a custom radio station in the Fallout 4 Creation Kit, from setting up the quest and radio records to getting it working in-game on custom radios, jukeboxes, and the Pip-Boy.

It also includes my enhanced random shuffle script, adapted from YouTuber HeartImpaled's original framework, available here[d2dmusic.blogspot.com].

The original script provides an excellent foundation, but I've expanded it with improved randomization and replay-avoidance logic to produce a more natural listening experience with fewer repeated tracks.

While there are many tutorials covering parts of the process, I found that most are either incomplete or out of date. After plenty of trial and error, countless hours watching community tutorials, and digging through the official Creation Kit documentation, I finally assembled a workflow that produces a fully functional radio station capable of efficiently shuffling music, DJ dialogue, advertisements, and other segments.

Rather than spending hours piecing information together from multiple sources, you can follow this guide from start to finish and have your own radio station up and running.

Let's get started!

Create Your Mod

1. Open the Creation Kit.
2. Select *File* > *Data*.
3. Double-click *Fallout4.esm* (the first entry) so it becomes the active master.
4. Wait for the data to finish loading.
5. Select *File* > *Save*, then give your mod a name.

Create the Quest

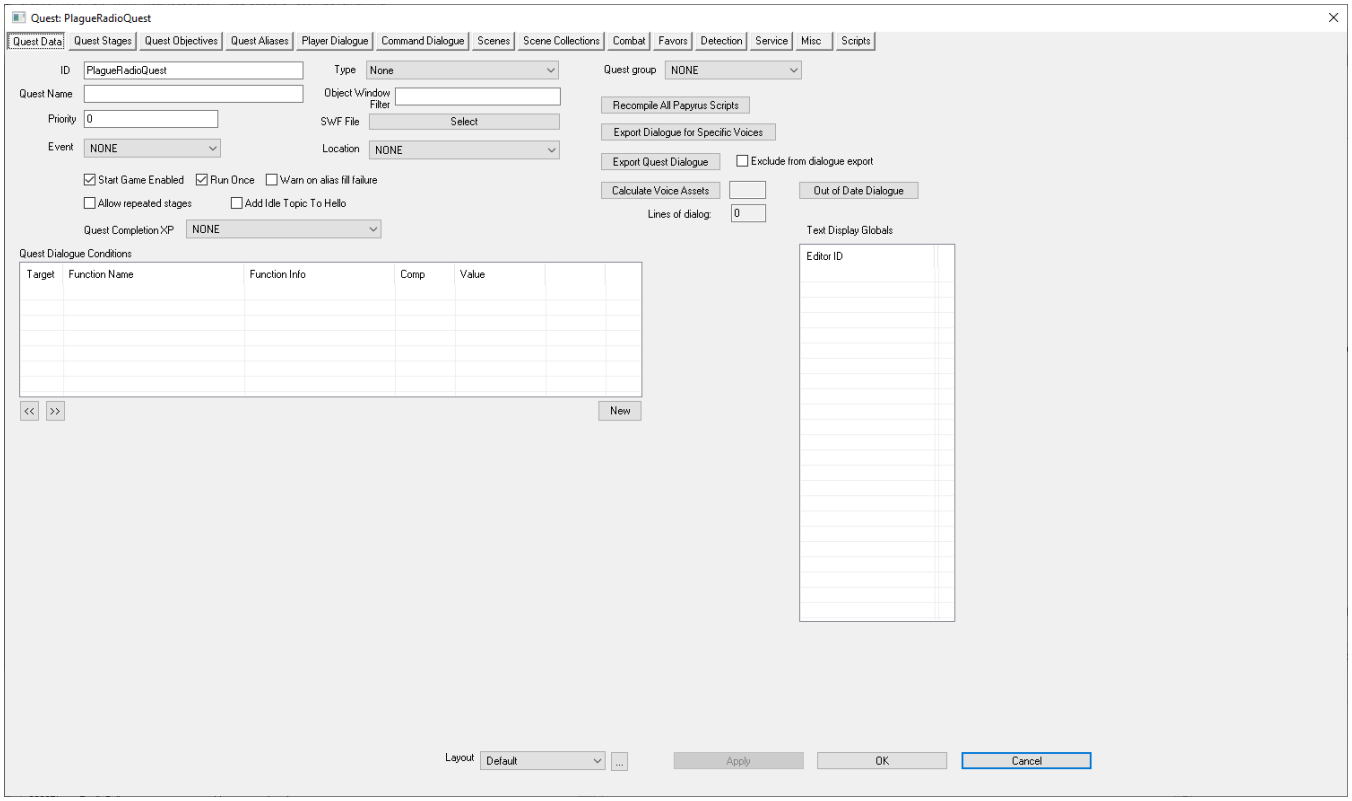
In the Object Window, navigate to:

Character > *Quest*

Right-click inside the object list and select *New*.

Give the quest a unique ID, for example:

MyRadioStationQuest

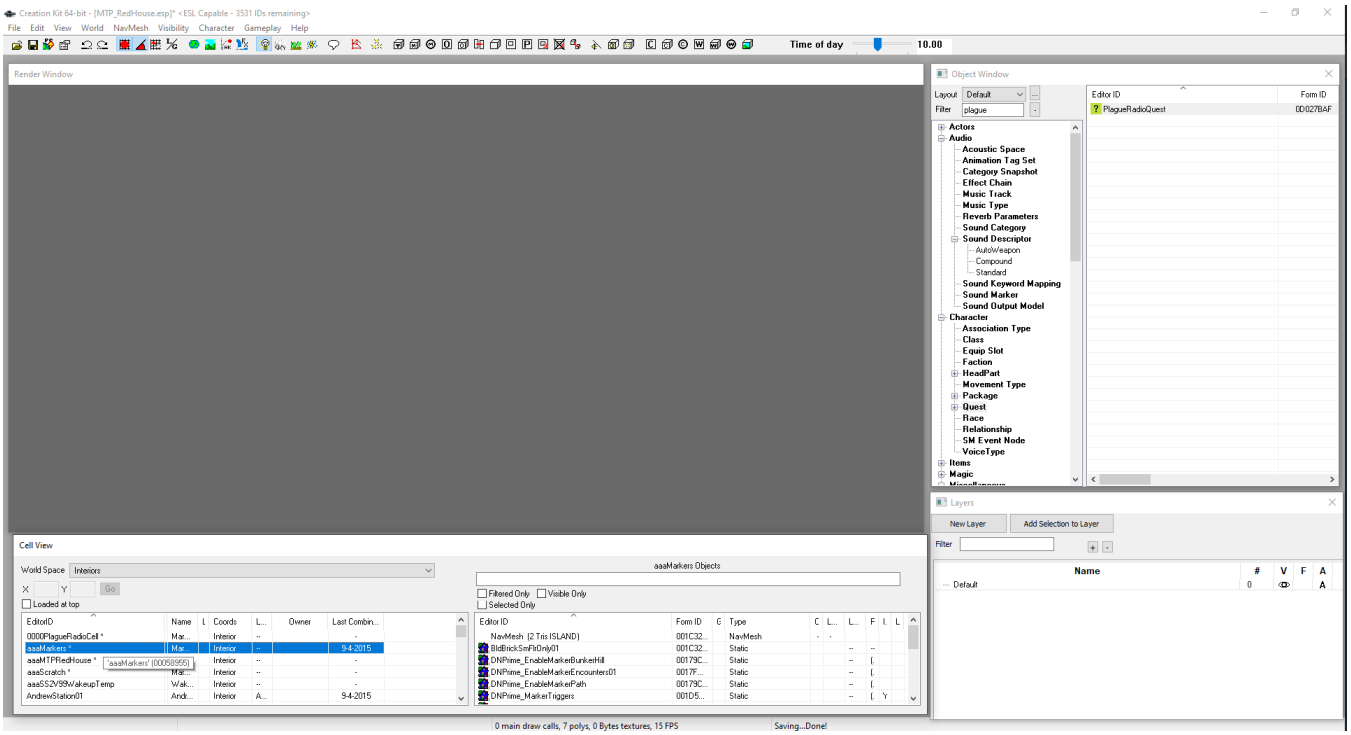


Click *OK*, then save your mod.

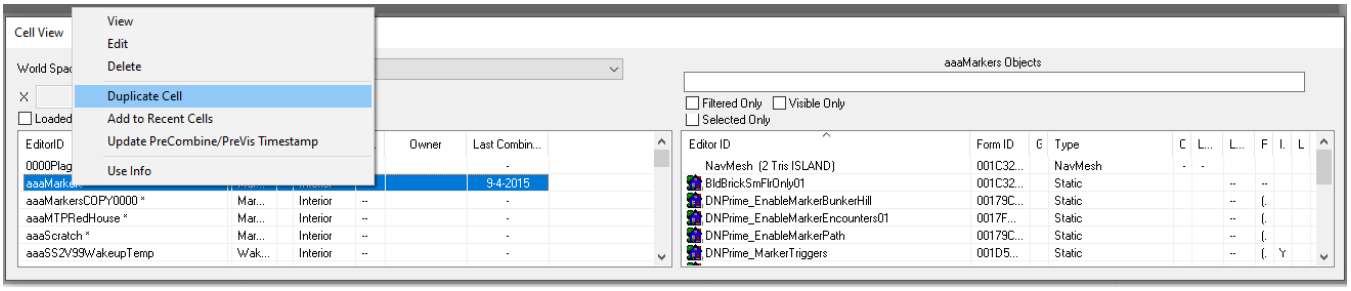
2) Create a Dedicated Cell (or Choose an Existing Location)

Now decide where you want your radio transmitter to exist. It can be placed almost anywhere in the Commonwealth, but I recommend creating a dedicated cell.

In the *Cell View* window:



Right-click *aaaMarkers* and select *Duplicate Cell*.

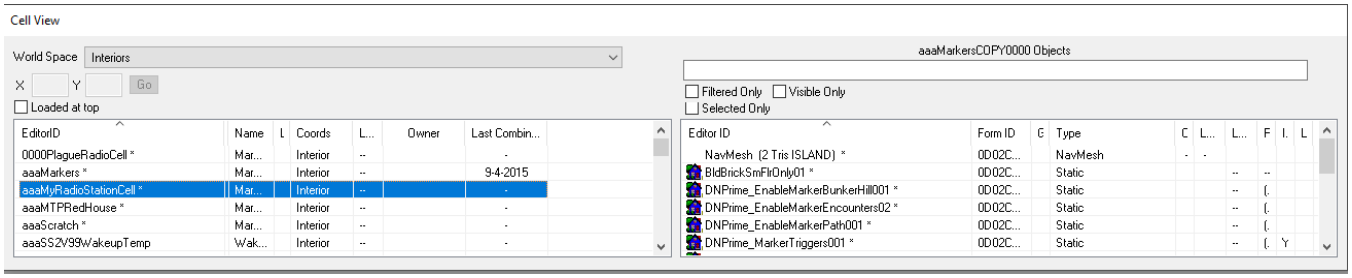


Rename the duplicate by *slow double-clicking* the new entry (*click, pause, click*) and giving it a unique ID.

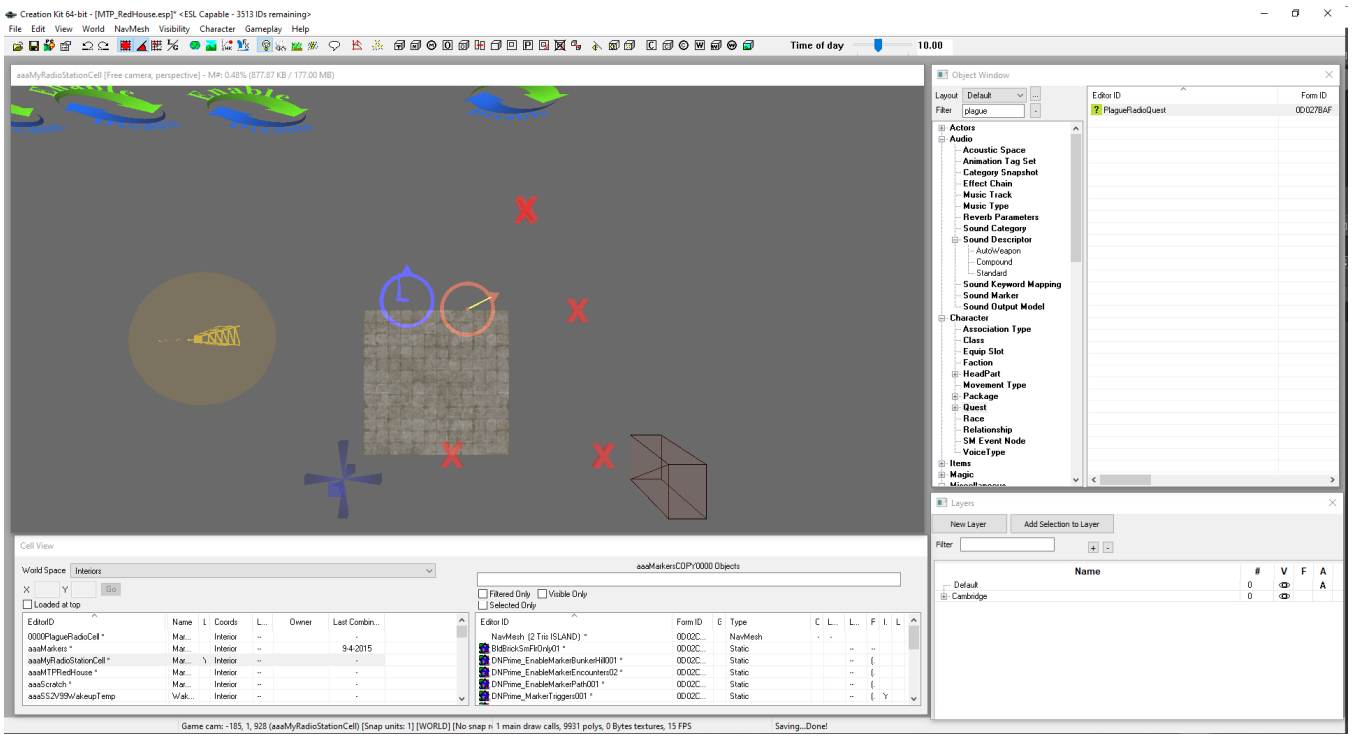
Use only letters and numbers here: no spaces, punctuation, hyphens, or underscores.

For Example:

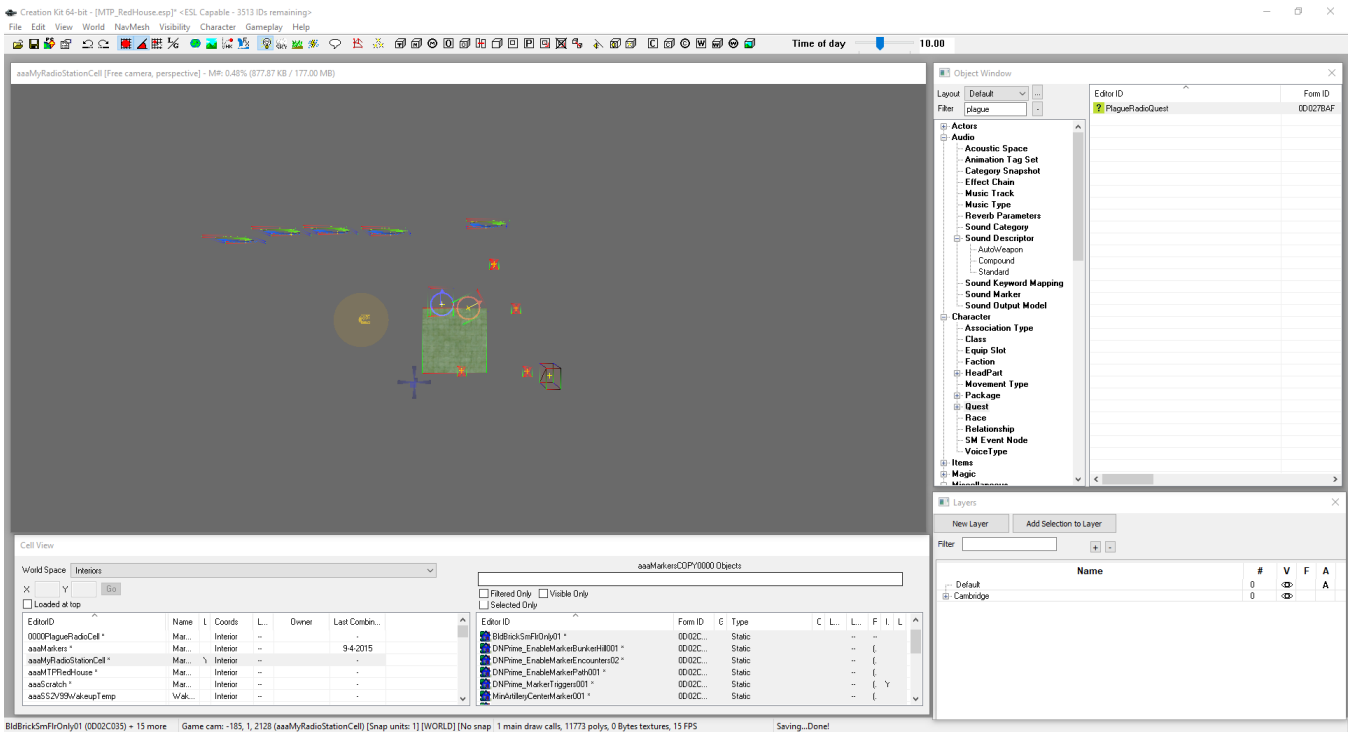
aaaMyRadioStationCell



Double-click your new cell to load it into the Render Window.



To clear the cell, right-click the *Layer* in the *Cell View* window and choose *Select All Loaded References*.



With everything selected, click the Render Window titlebar to give it focus, then press *Delete*.

Also, you can zoom out, drag a selection box around everything manually, and press Delete. Since this is a new duplicated marker cell, you don't need to worry about precombines.

Your new cell should now be completely empty and ready to hold your radio transmitter.

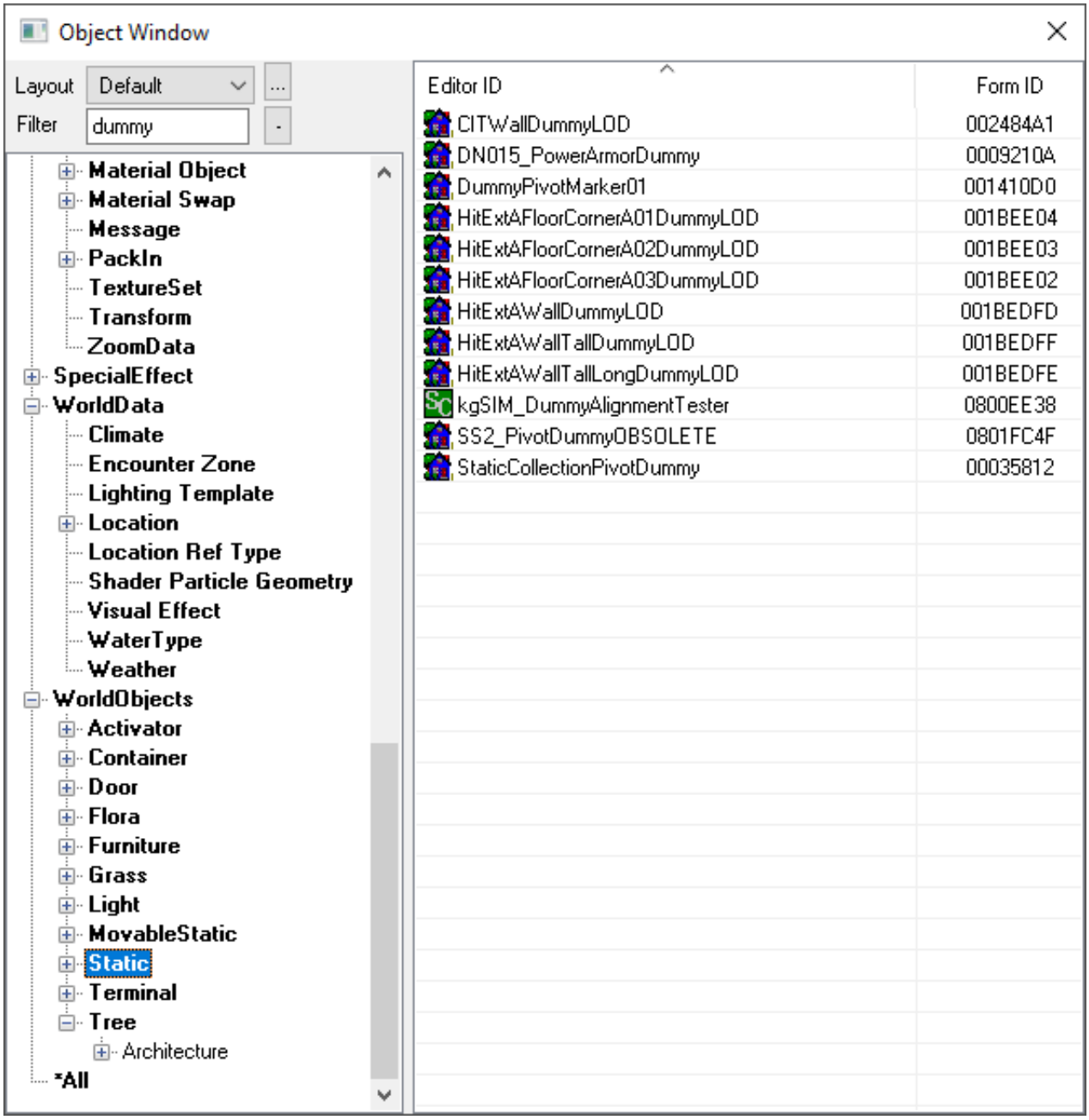
Save your mod before continuing.

3) Create Your Radio Transmitter

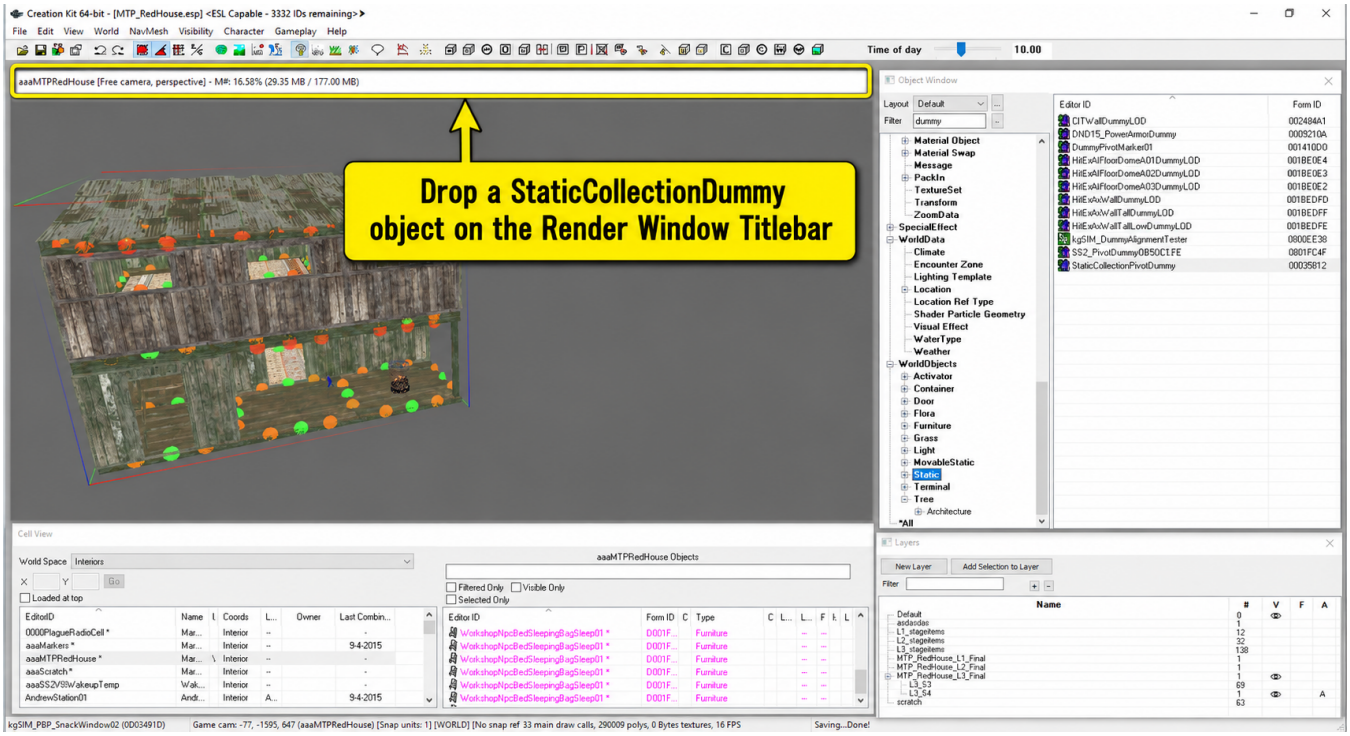
With your radio station cell still loaded, locate the *Filter box* in the *Object Window* and search for:

dummy

In the left pane, expand *WorldObjects* and select *Static*.



In the right pane, find *StaticCollectionPivotDummy* and drag it onto the titlebar of the Render Window.



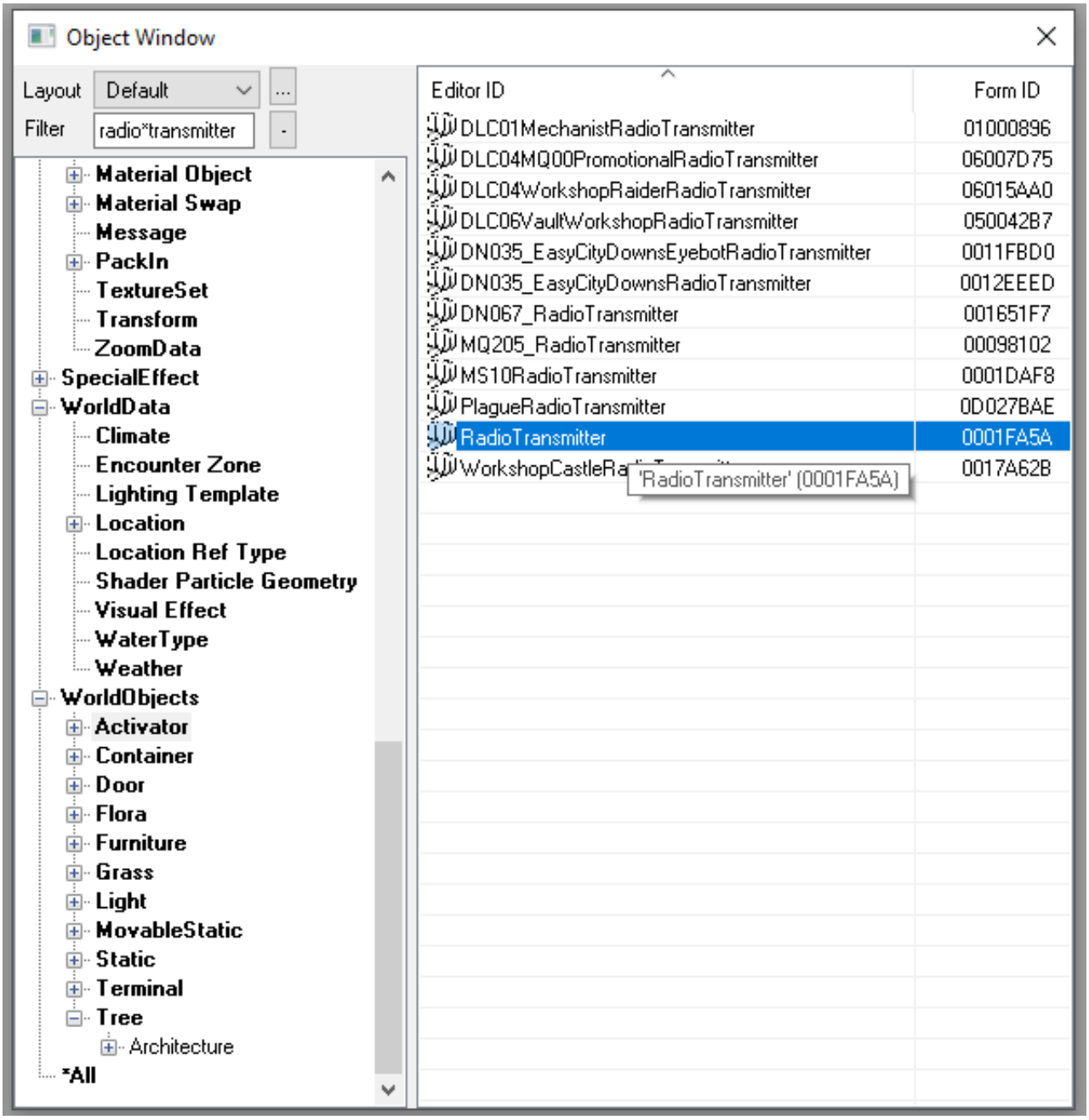
This isn't required, but it makes navigating the Render Window much easier.

Next, clear the filter and search for:

*radio*transmitter*

In the left pane, select *WorldObjects* > *Activator*.

Locate *RadioTransmitter* in the right pane and double-click it to create a new form based on the existing transmitter.

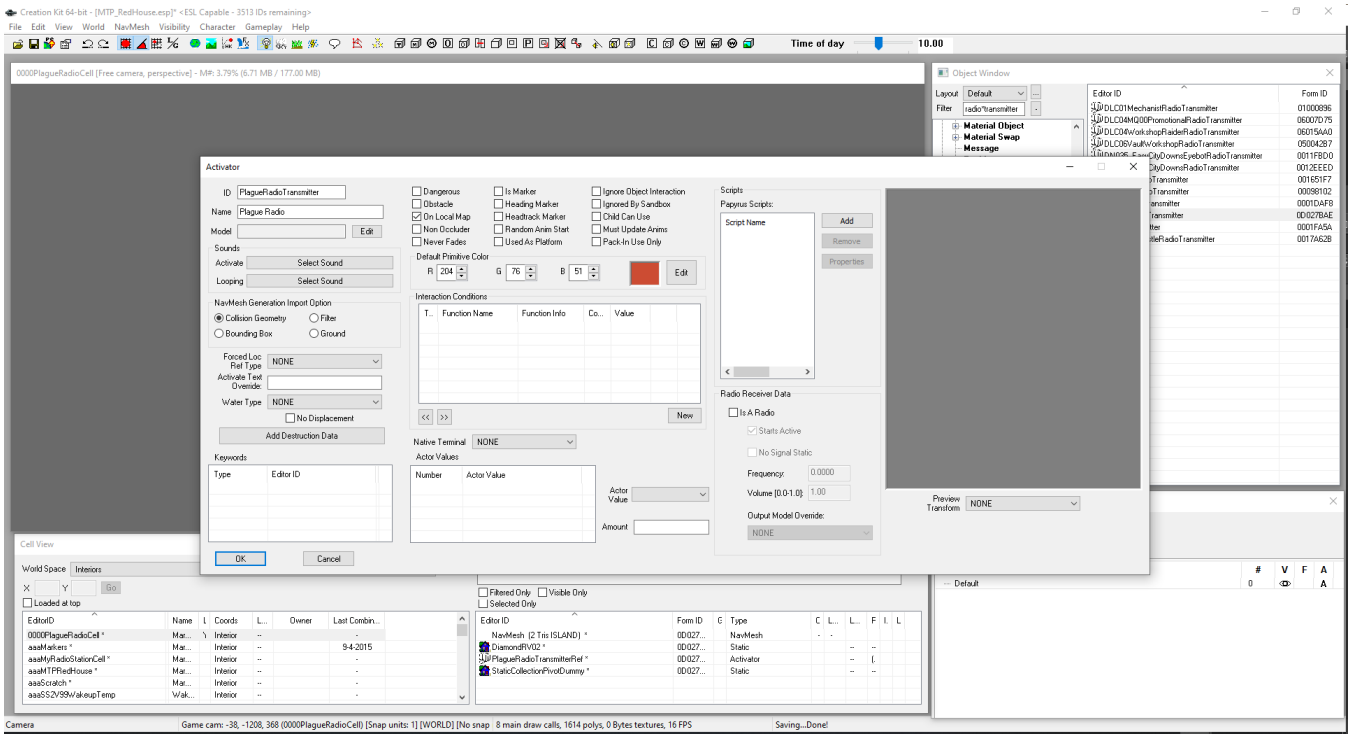


Give the new transmitter a unique *ID*, for example:

MyRadioTransmitter (how clever)

Then change the Name field from the default ("Emergency...") to the name of your radio station, for example:

My Radio Station



Click *OK*, then choose *Yes* when prompted to create a new form.

You now have your own custom radio transmitter that can be linked to your radio station without modifying any vanilla Fallout 4 records.

4) Create the Pip-Boy Radio Station Name Message

In the *Object Window*, expand *Miscellaneous* and select *Message*.

Right-click in the right pane and choose *New*.

Give the message a unique *ID* related to your radio station.

For the *Title* field, enter the name you want your station to display in the Pip-Boy's *Radio* tab.

Click *OK* to create the message.

Later in the guide, you'll assign this message to the radio station so the Pip-Boy displays your custom station name.



Save your mod before continuing.

5) Place Your Radio Transmitter

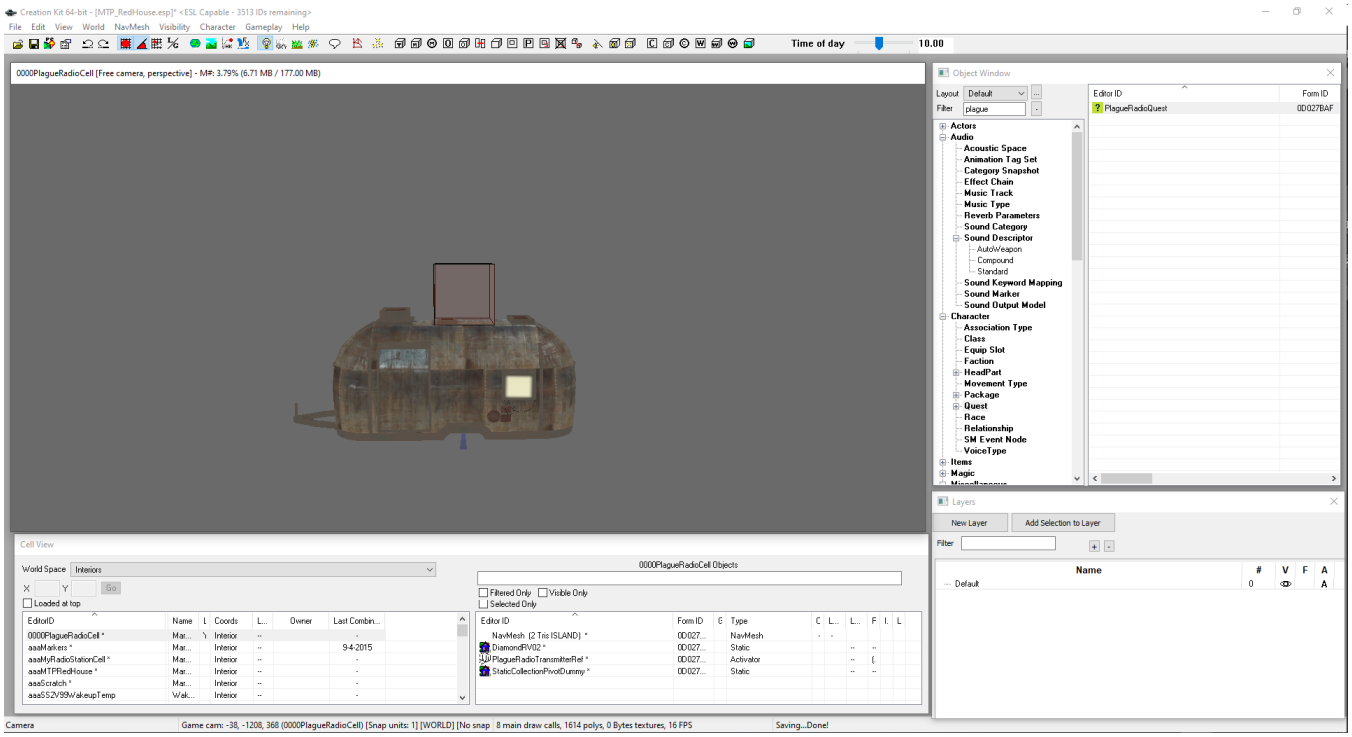
Return to your radio station cell if you're not already there.

In the *Object Window*, navigate to *World Objects > Activator* and use the *Filter* box to find the custom transmitter you created earlier.

You can make this your own by building a station from prefabs or buildings pieces (I used the DiamondCityRv02 static), a radio tower, satellite station - whatever you like. Just remember that your static pivot dummy is the "center" of your cell and use it to your advantage for alignment and navigating.

Drag the transmitter into the Render Window and drop it where you want it. Fine tune the placemetn if needed.

If you don't see either the transmitter (a wireframe cube) or the *StaticCollectionPivotDummy*, markers are probably hidden. Press *M* to toggle marker visibility.



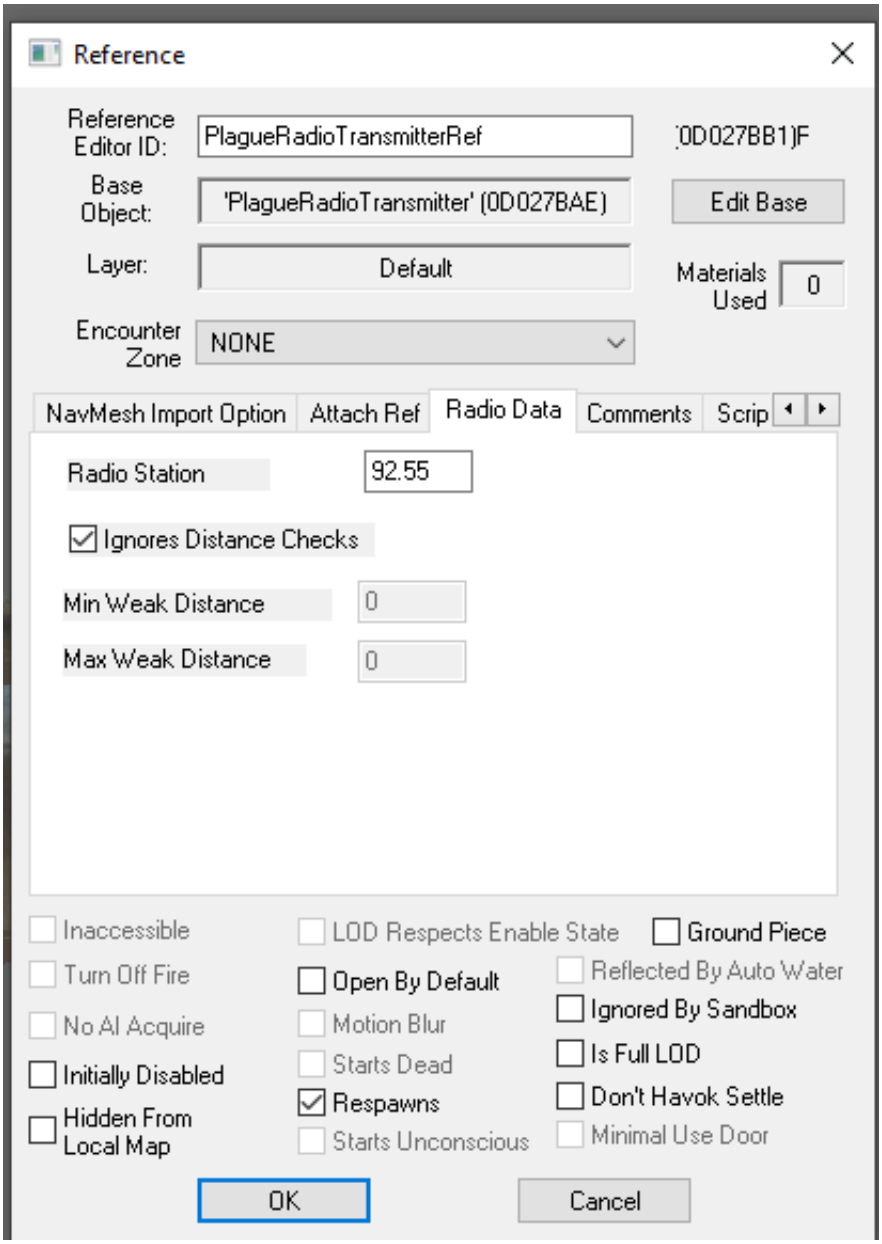
Double-click the transmitter to open its Reference window.

Give the reference a unique *Reference ID*, for example:

"*MyRadioStationRef*"

Open the *Radio Data* tab and configure the following:

- Frequency: **92.55** (currently a safe frequency to use)
- Enable *Ignore Distance Checks*



A few notes about radio frequencies:

- Low frequencies will not appear in the Pip-Boy Radio menu.
- Before choosing a frequency, check for conflicts with existing stations.
- See: fallout 4 radio frequency list here[fallout.wiki]

Click *OK* to save the reference.

Save your mod before continuing.

6) Quest Settings and Quest Alias

Now under *Character* > *Quest*, use the *Filter* box to find the quest you created earlier and open it.

On the *Quest Data* tab, make sure both of the following are checked:

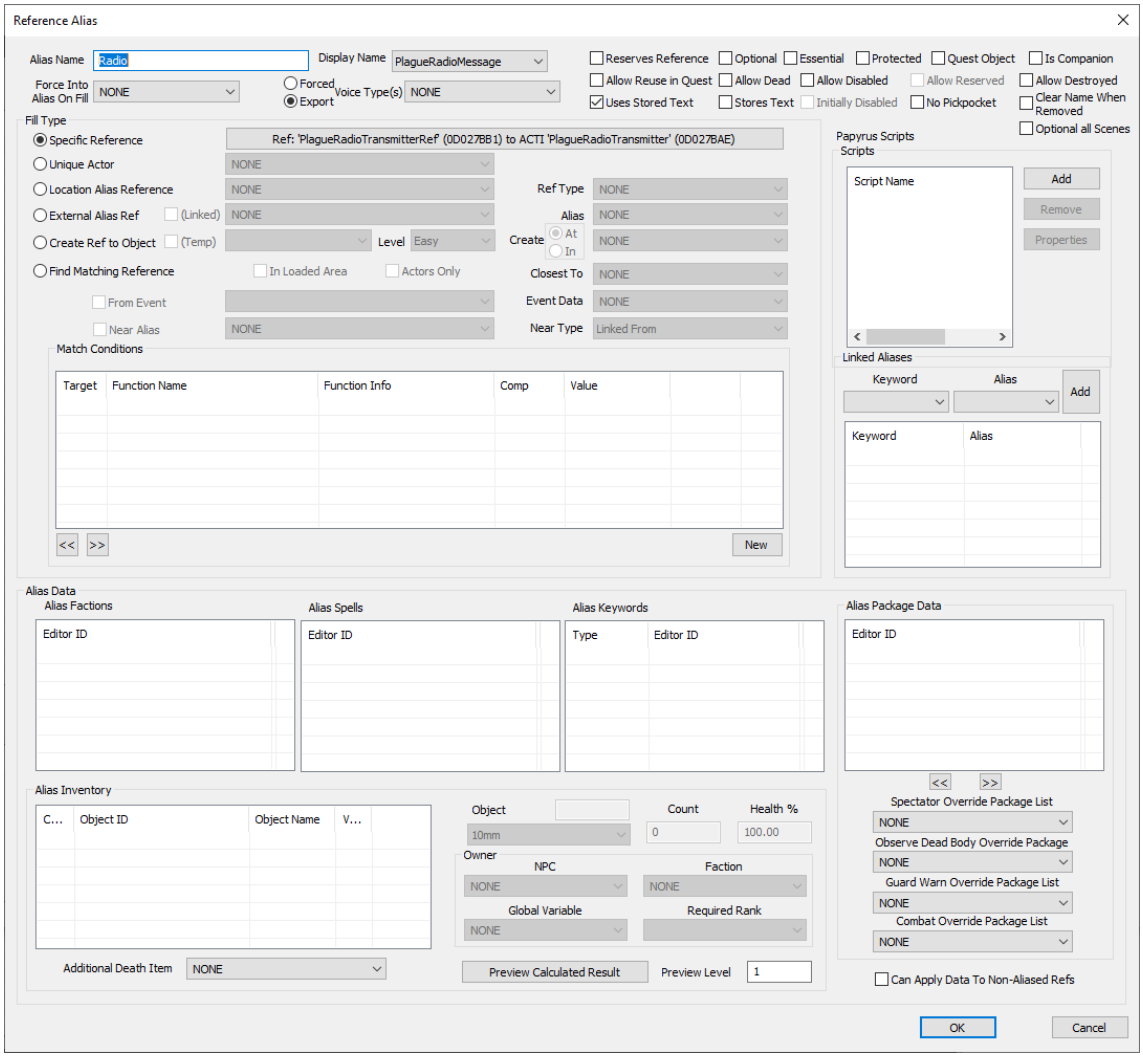
- *Start Game Enabled*
- *Run Once*

Next, open the *Quest Aliases* tab.

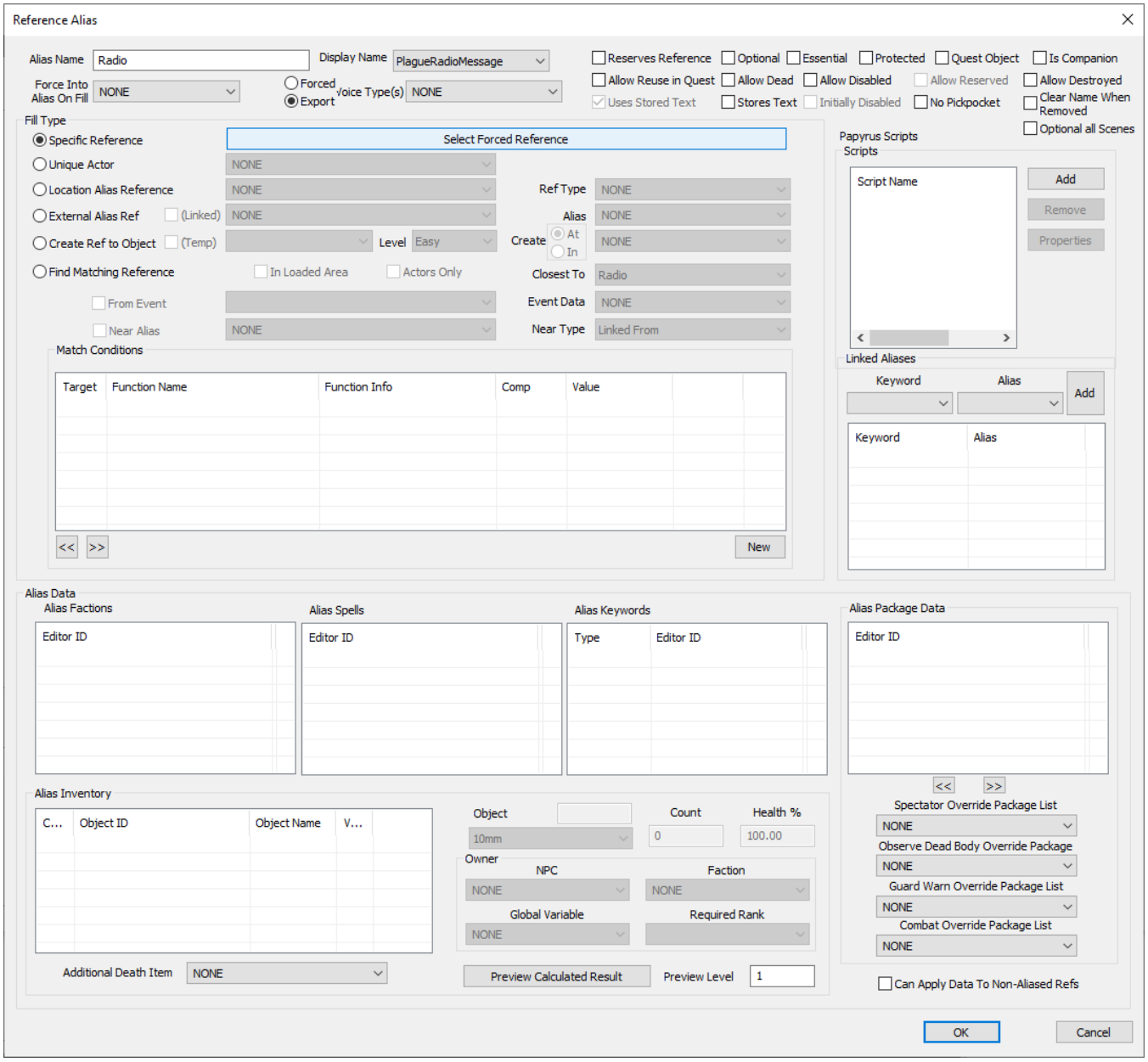
Right-click in the alias list and choose *New Reference Alias*.

In the new alias window:

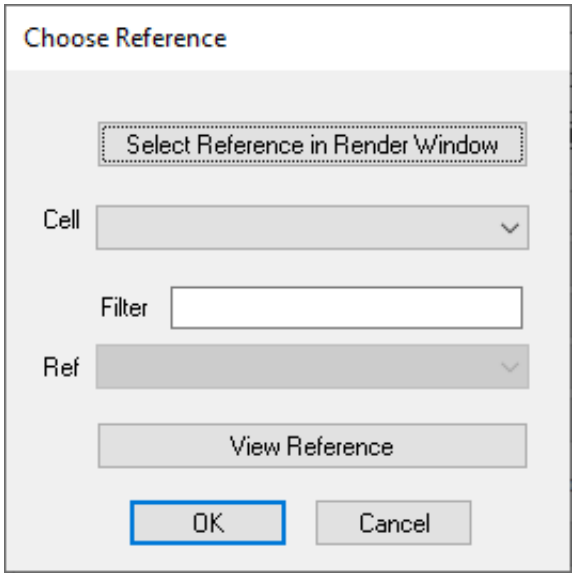
- Set *Alias Name* to "*Radio*".
- Click the dropdown next to *Display Name* and select the message you created earlier for your station's Pip-Boy name.
- The *Uses Stored Text* checkbox should be checked (and likely diabled).



Select the *Specific Reference* radio button.



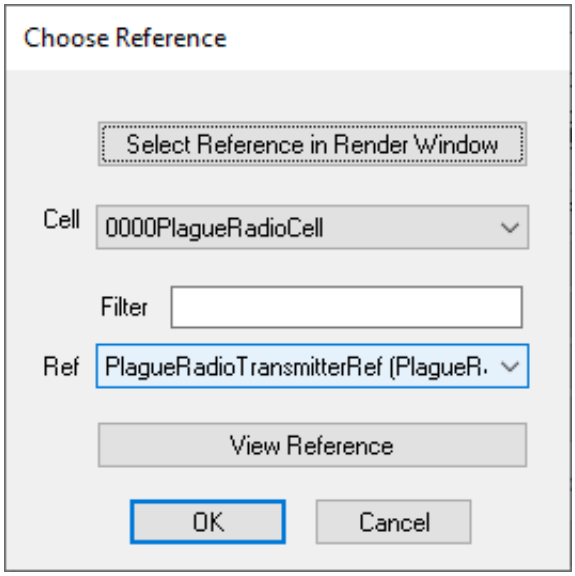
Click *Select Forced Reference*, then choose *Select Reference in Render Window*.



Your cursor will change to a crosshair and the Render Window will come to the front.

Double-click your radio transmitter.

Before clicking *OK*, make sure your radio transmitter is actually the reference that was populated. If it isn't, repeat the process.



Click *OK* to close the alias window.

Click *OK* again to close the Quest Aliases tab.

Finally, click *OK* one last time to close the quest.

Save your mod before continuing.

7) Sourcing and Converting Songs

Prepare Your Audio Files

Before adding music to your station, a quick note about copyright.

If you're only making the mod for yourself, use whatever music you like.

If you intend to distribute your mod, you'll need to satisfy the copyright gods by using music that's in the public domain or that you have permission to distribute. As a general rule, works over 100 years old are safe, but always verify the copyright status yourself.

Fortunately, there is plenty of great music from the era that ifits with Fallout Lore. Blues, Jazz, Ragtime, Big Band, Lounge, and music from the Great Depression and Prohibition all fit remarkably well.

Now for the technical part.

THe Fallout 4 engine expects your audio to be in a specific format before it can be converted into the game's native audio format.

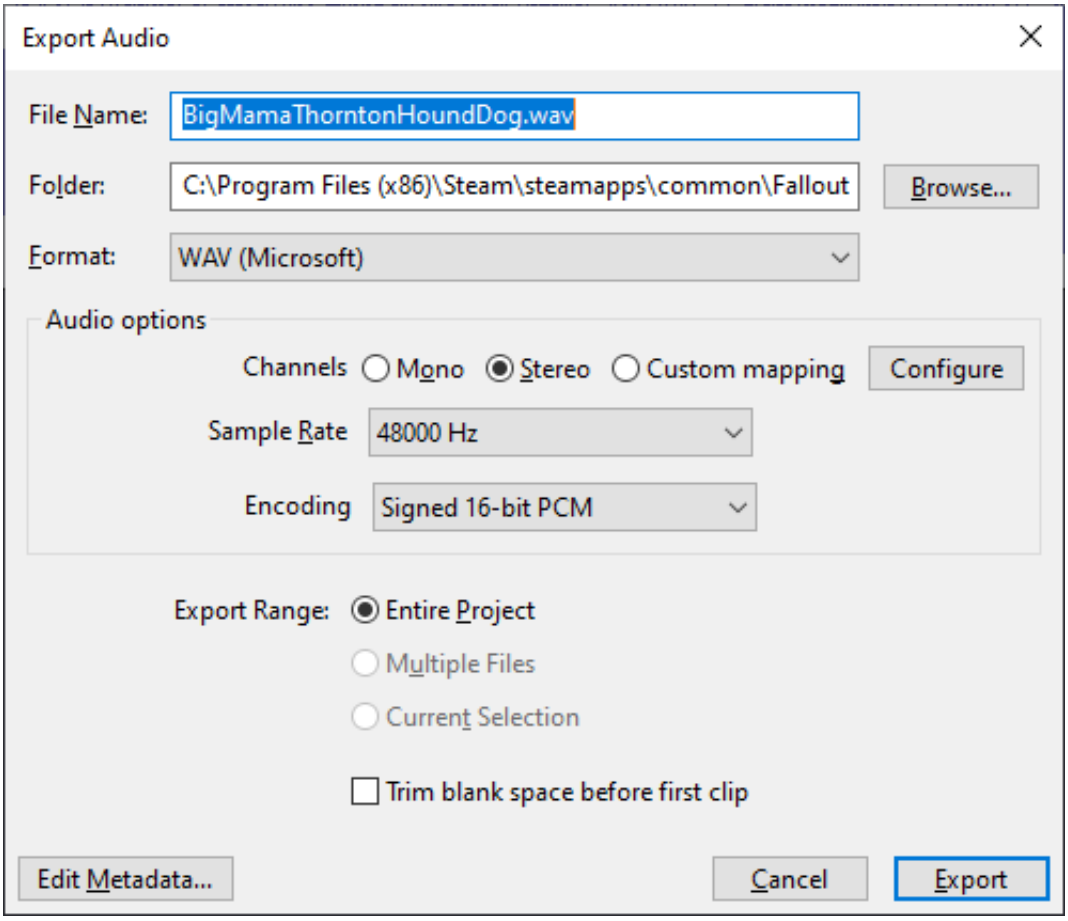
The easiest way to do this is with Audacity (download here[www.audacityteam.org]). I can't guarantee other audio editors will work, but any tool capable of exporting a WAV with the following settings should be fine.

Export each audio file using:

- Folder: *[Your Fallout 4 Install]\Data\Sound\fx\mus\radio[YourModName]*
- Format: **WAV (Microsoft)**

Audio Options:

- Channels: Stereo
- Sample Rate: 44100 Hz
- Encoding: Signed 16-bit PCM



Click *Export*. Once the export completes, close Audacity. There's no need to save the Audacity project unless you plan to edit the audio later.

Repeat this process for every music track, DJ segment, advertisement, or other audio clip you want to include in your station.

When you're finished, your converted WAV files should be located at:

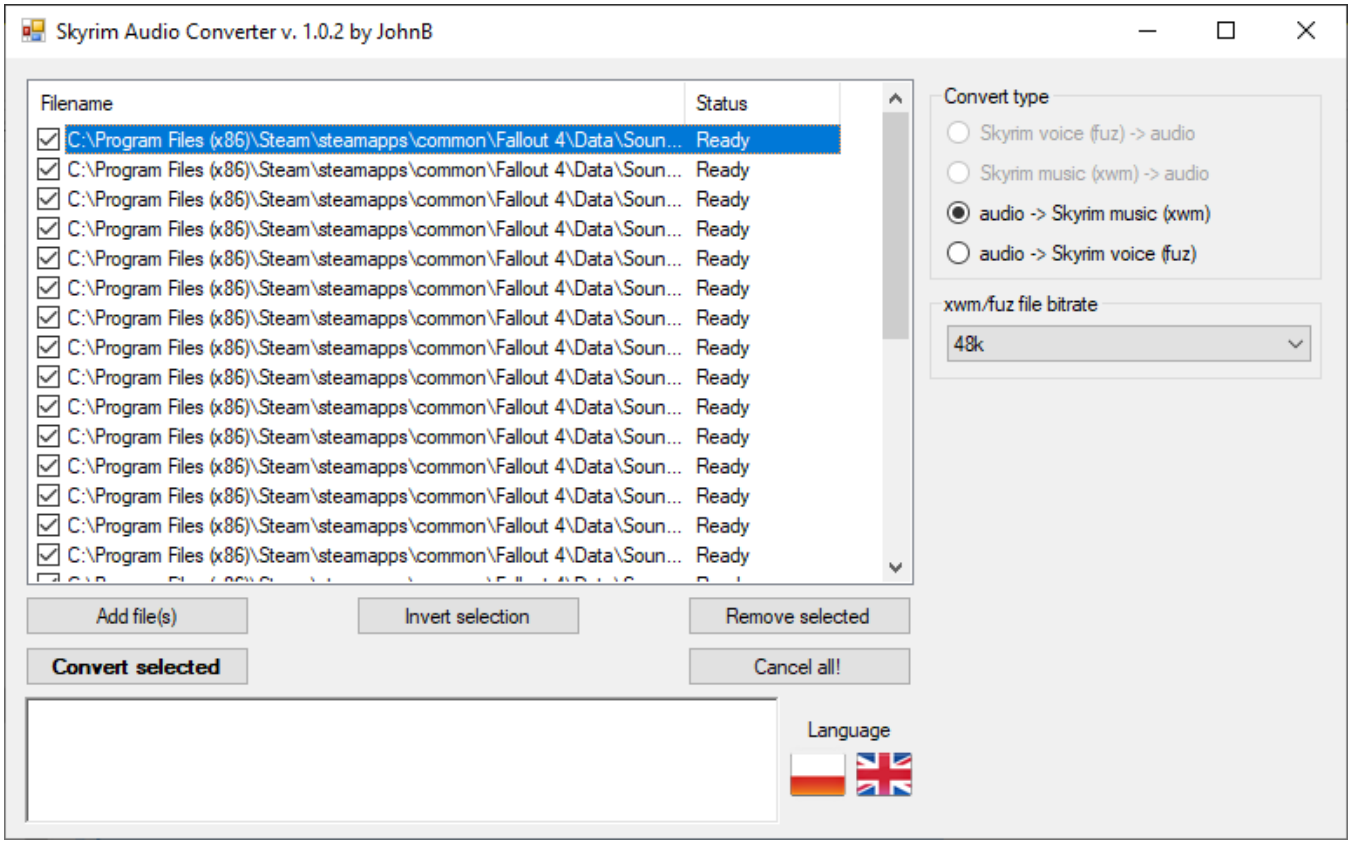
Data\Sound\fx\mus\radio[YourModName]

Important: The folder name should **not** end with ".esp". If it does, rename the folder so it matches your mod name without the extension.

Next, you'll need the Skyrim Audio Converter (download here[www.nexusmods.com]).

Despite the name, it works perfectly for Fallout 4.

Simply drag all of your WAV files into the converter and click *Convert Selected*.



By default, the converter writes the resulting XWM files to the same folder as the source WAVs, so if you exported everything to the correct directory, your XWM files will already be in the right place.

Keep the original WAV files for now. You'll need them later when importing your audio into the Creation Kit.

8) Importing songs

Import Your Songs and Audio Clips

In the *Object Window*, navigate to *Audio > Sound Descriptor*.

You can either create a new Sound Descriptor, or use an existing radio descriptor as a template.

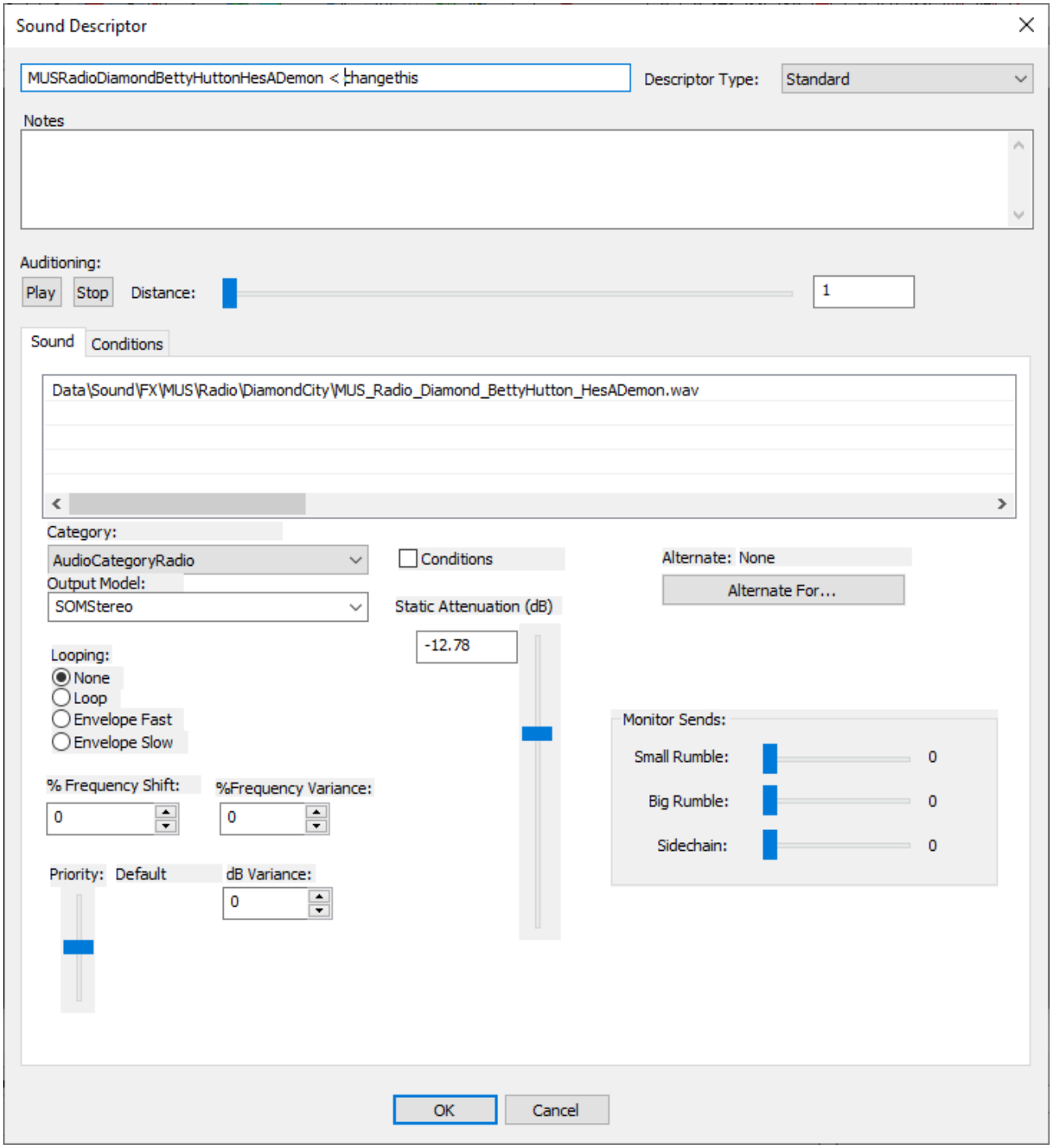
I recommend using a template.

Filter for:

*mus*diamond*

Double-click the first result, then give it a unique ID, for example:

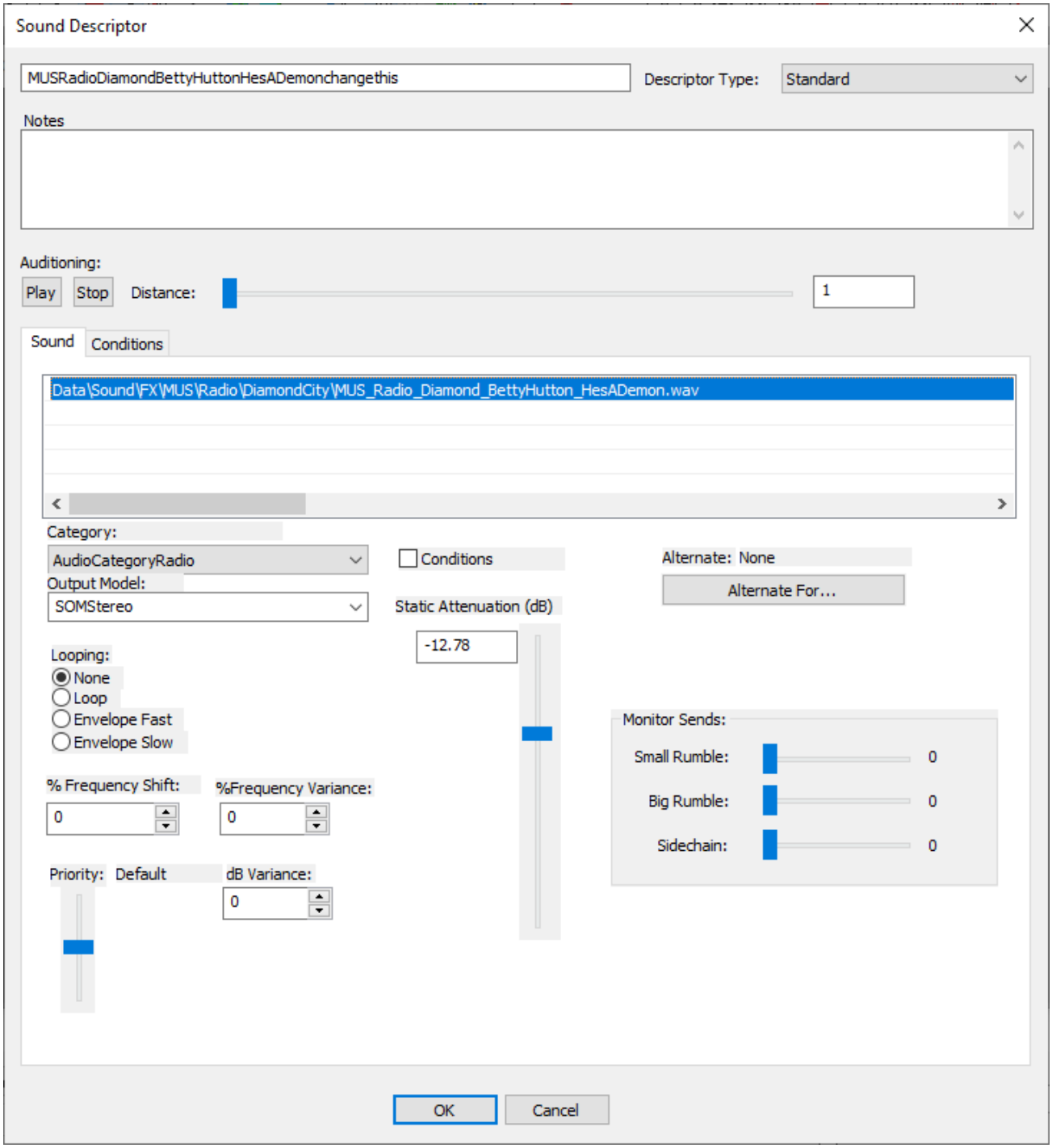
"*MUSMyTrack01*"



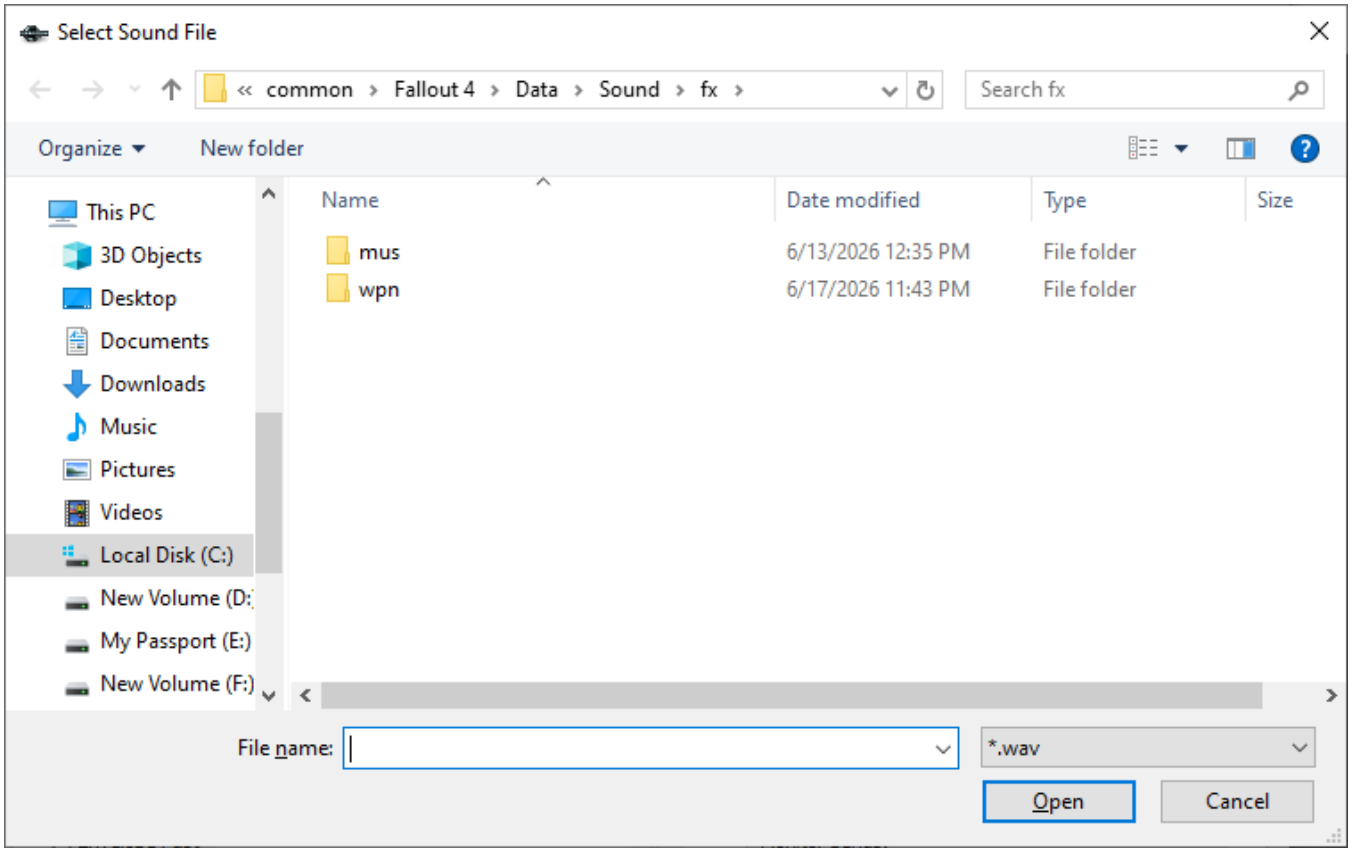
Open the *Sound* tab.

If you created your descriptor from an existing one, right-click the sound entry and choose *Delete*.

Then right-click again and choose *New*.



A file browser will open.



Browse to:

Data\Sound\fx\mus\radio[YourModName]

Select the corresponding **.wav** file for your first track.

Leave the remaining settings at their default values. They are already configured for radio playback.

Repeat this process for every music track, DJ segment, advertisement, station ID, or other audio clip you want your station to play.

Save your mod frequently while importing audio. The Creation Kit is known to crash, and frequent saves can prevent you from losing a lot of work.

Once you've created Sound Descriptors for all of your audio files, you can safely delete the original **.wav** files.

Keep the **.xwm** files. Those are the files Fallout 4 actually plays in-game.

9) Make a Scene

Open your quest and select the *Scenes* tab.

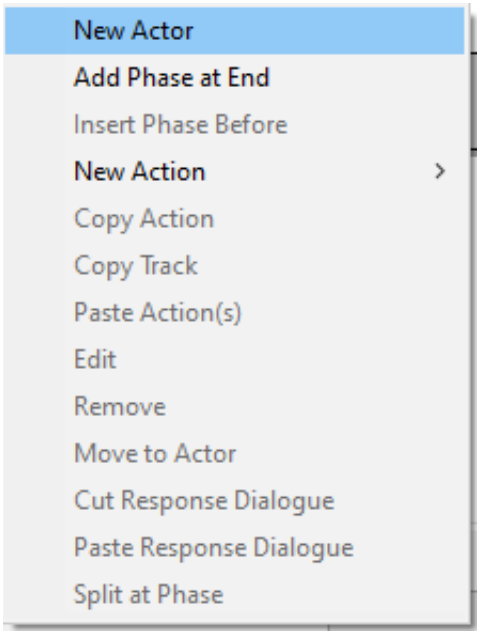
In the upper-left panel, right-click and choose *New*.

Give the scene a unique ID, for example:

"*MyRadioQuestScene*"

In the large center pane, right-click and choose:

New Actor



Select the station alias you created earlier.

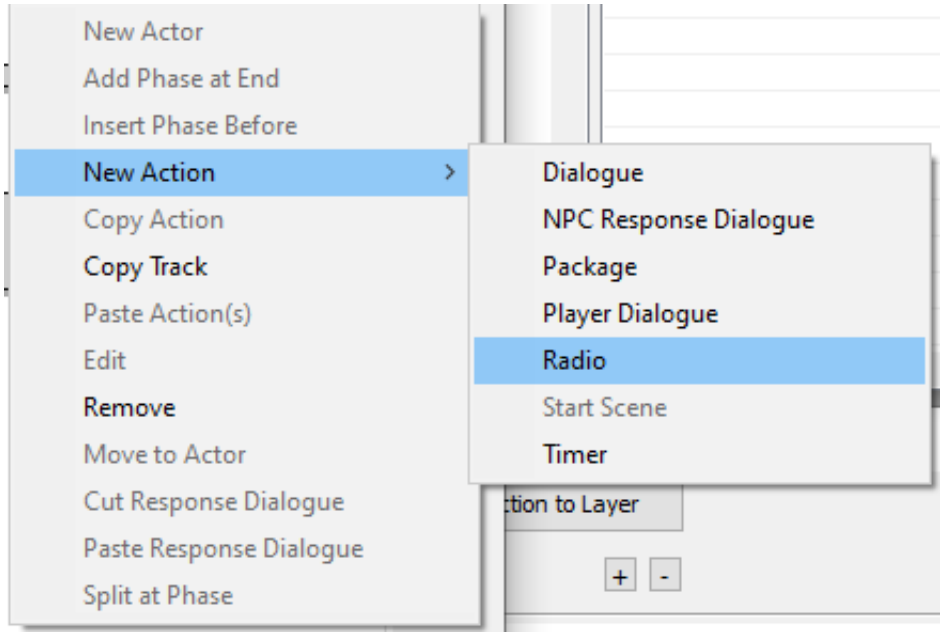
The actor should now appear in the scene as *Radio*.

Right-click where the word *Radio* appears and choose:

Add Phase at End

Right-click the new phase and choose:

New Action > Radio



Give the action a unique ID. I like using the artist and song title, for example:

"*[SongArtist][SongTitle]*" Like:
JimmiHendrixRedHouse

Click *Select Sound* and choose the Sound Descriptor you created for your first track.

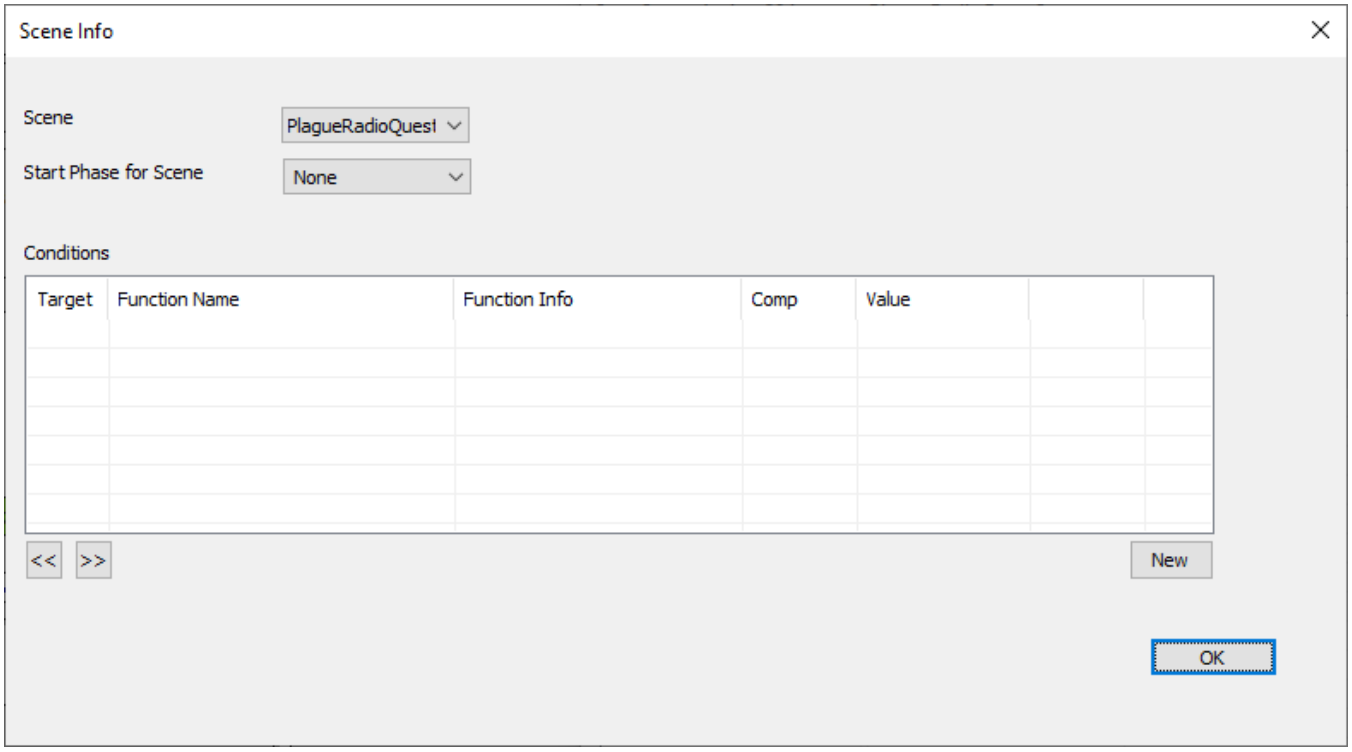
Repeat this process until you've added every song, DJ segment, advertisement, or other audio clip to the scene.

If you're building a large playlist, close the scene and quest occasionally, then save your mod. It's much better to lose a few entries than an entire playlist if the Creation Kit decides it's had enough.

Once you've added your final track, you'll need to make the scene loop.

After creating the last *Radio* action:

- Right-click and choose *Add Phase at End*.
- Right-click the new phase and choose *New Action > Start Scene*.
- Right-click the blank action area (you don't need to name it or change any options).
- Choose *New*.
- Select your radio scene (for example, *MyRadioQuestScene*) from the *Scene* drop-down list.



Click *OK*.

You should now see an arrow pointing from the final action back to the first action in the scene.

If the arrow isn't there, double-check the previous steps before continuing.

Save your mod before continuing.

10) The Shuffle Script

Add the Shuffle Script

Open your quest and select the *Scripts* tab.

Click:

- *Add*
- *New*

Configure the new script as follows:

- Name: *PlagueRadioShuffler*
- Namespace: Leave blank
- Extends: *Quest*
- Check the *Conditional* box

Click *OK*.

When the Properties window opens, click *Cancel*. We'll come back to that later.

Back on the Scripts tab, right-click your new script and choose *Edit Source*.

Paste in the shuffle script below, adapted from YouTuber HeartImpaled's original randomizer script, available here[d2dmusic.blogspot.com].

Important:

The first line of the script must match the name you gave your script in the Creation Kit.

For example, if your script is named:

PlagueRadioShuffler

then the first line of the script should be:

```
Scriptname PlagueRadioShuffler extends Quest Conditional
```

Paste the complete script from below:
Updated source review

This export uses the current PlagueRadioShuffler.psc from Data\Scripts\Source\User in the local Fallout 4 installation. Each shuffled bag contains every track once. With more than one track, the first track of the next bag cannot immediately repeat the last track of the previous bag. There is no fixed recent-history window or iNoRepeatWindow property.

The unchanged source compiled successfully with the official Fallout 4 Papyrus compiler 2.8.0.4: 1 succeeded, 0 failed. A new PEX was produced separately, without replacing the existing loose PlagueRadioShuffler.pex, which contains the older recent-history algorithm. The new PEX has not been installed or tested in-game, and scene-fragment compatibility remains unverified. Recompile the matching scene fragments before testing. Which compiled script the game actually loads was not verified.

```
Scriptname PlagueRadioShuffler extends Quest Conditional

Int Property iCurrentTrack = -1 Auto Conditional
{The track currently selected for playback.}

Int Property iPrevTrack = -1 Auto Conditional
{The previously played track.}

Int Property iPrevPrevTrack = -1 Auto Conditional
{The track played before the previous track.}

Int Property iTrackCount = 0 Auto Conditional
{Total number of tracks in the playlist. Track IDs are expected to range from 0 through
iTrackCount - 1.}

Int[] _bag
Int _bagIndex = 0
Int _lastPlayedTrack = -1


Function InitializeBag()

    If iTrackCount <= 0
        _bag = new Int[0]
        _bagIndex = 0
        Return
    EndIf

    _bag = new Int[iTrackCount]
```

```
    Int i = 0

    While i < iTrackCount
        _bag[i] = i
        i += 1
    EndWhile

    i = iTrackCount - 1

    While i > 0

        Int j = Utility.RandomInt(0, i)

        Int temp = _bag[i]
        _bag[i] = _bag[j]
        _bag[j] = temp

        i -= 1

    EndWhile

    If iTrackCount > 1 && _lastPlayedTrack >= 0

        If _bag[0] == _lastPlayedTrack

            Int swapIndex = Utility.RandomInt(1, iTrackCount - 1)

            Int temp = _bag[0]
            _bag[0] = _bag[swapIndex]
            _bag[swapIndex] = temp

        EndIf

    EndIf

    _bagIndex = 0

EndFunction


Int Function DrawNextTrack()

    If _bag.Length != iTrackCount || _bagIndex >= _bag.Length
        InitializeBag()
    EndIf

    If _bag.Length <= 0
        Return -1
    EndIf

    Int chosen = _bag[_bagIndex]

    _bagIndex += 1

    Return chosen

EndFunction


Function Randomize()

    If iTrackCount <= 0
        Return
```

```
    EndIf

    Int nextTrack = DrawNextTrack()

    If nextTrack < 0
        Return
    EndIf

    iPrevPrevTrack = iPrevTrack
    iPrevTrack = iCurrentTrack
    iCurrentTrack = nextTrack

    _lastPlayedTrack = nextTrack

EndFunction
```

When you're finished, press *Ctrl + F7*, or select:

Build > Compile

If everything was entered correctly, the script should compile without any errors.

If the compile fails, first check that the script name matches the one you created in the Creation Kit.

If that's correct, compare your script against the version above for any typing mistakes before trying again.

11) Set Script Properties

Set the Script's Properties

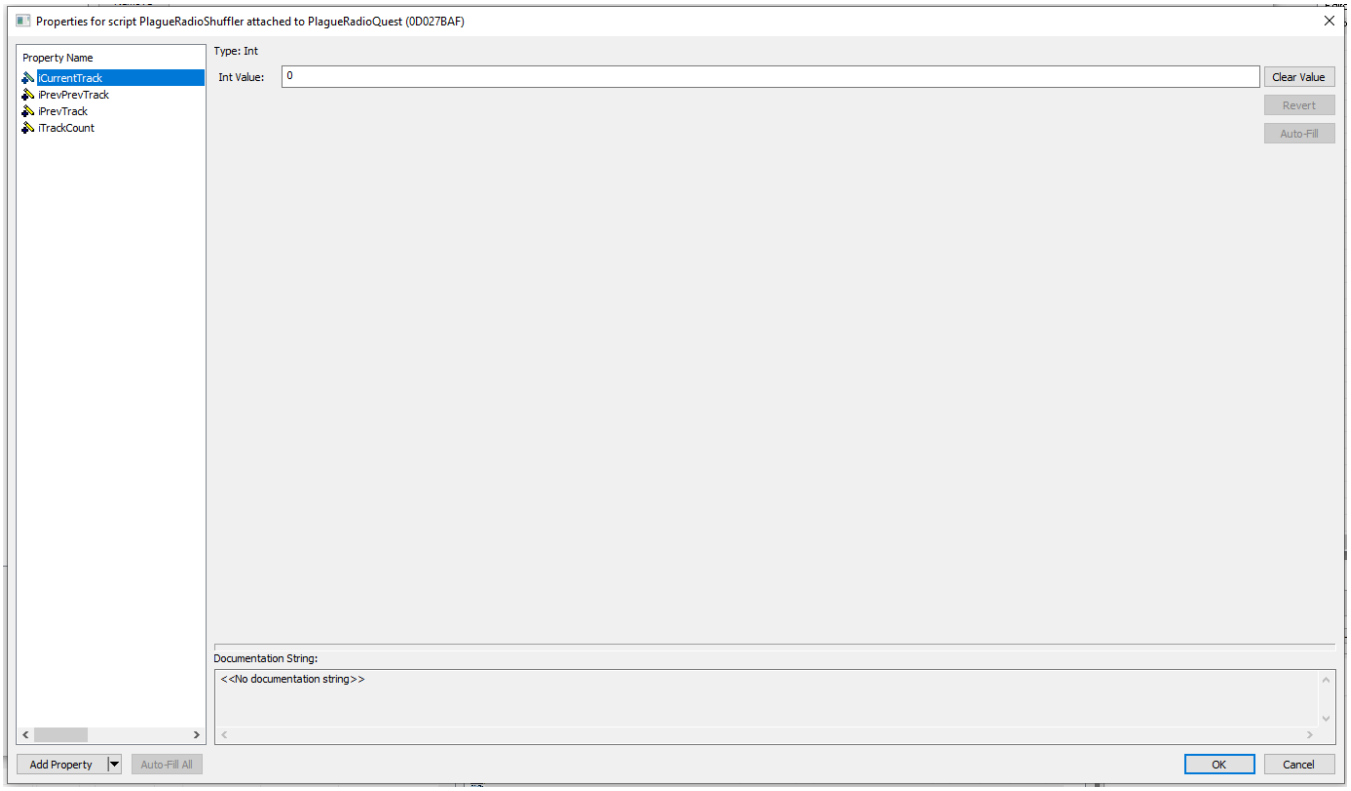
Before the shuffle script can work, you'll need to configure its properties.

Open your quest, go to the *Scripts* tab, right-click your shuffle script, and choose *Edit Properties*.

Set the following properties to:

- *iCurrentTrack*: -1
- *iPrevTrack*: -1
- *iPrevPrevTrack*: -1

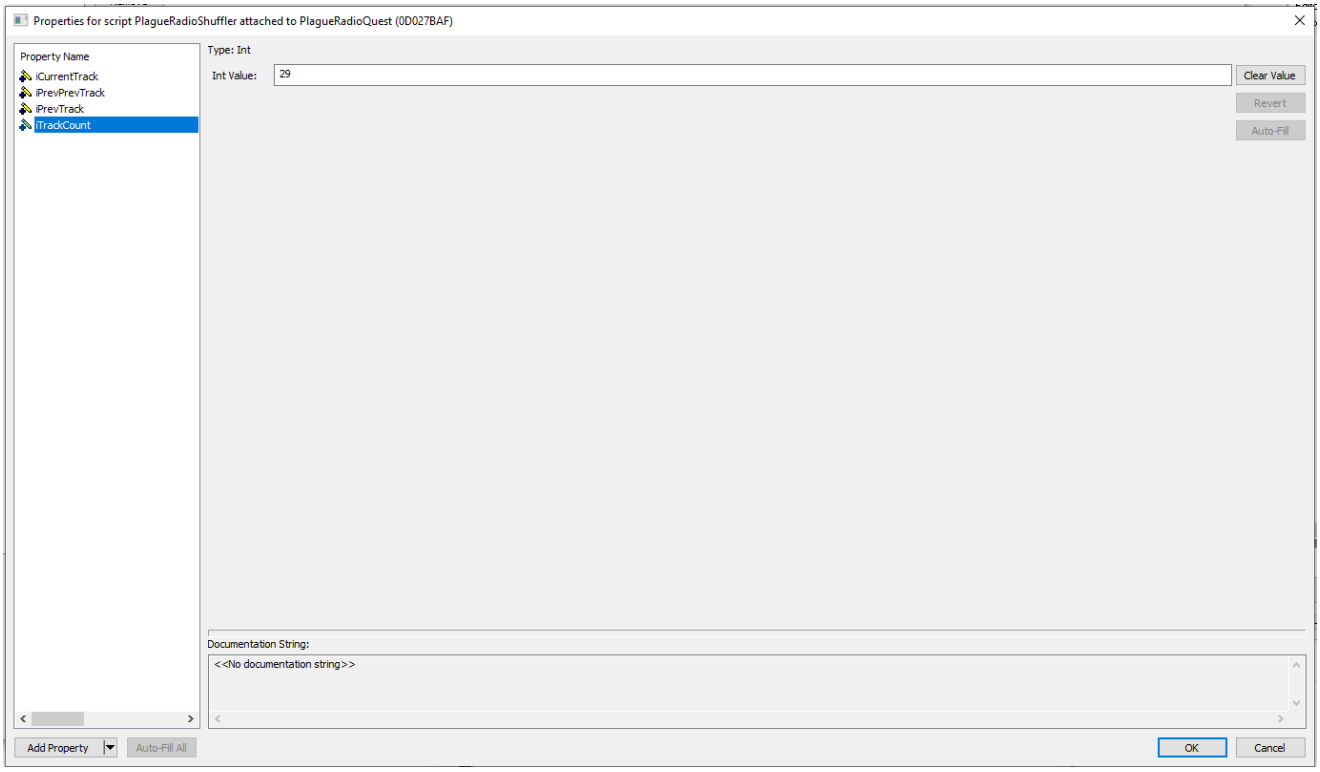
Keep these three track-history values at the source defaults of -1. Set iTrackCount to a positive number that matches the actual number of audio entries. The current script has only these four conditional properties; do not add iNoRepeatWindow. The original screenshots may show older names or values, so follow the updated text above.



Next, set:

- *iTrackCount*: The total number of audio tracks in your radio scene.

For example, if your station has 29 total tracks, your *iTrackCount* should be set to **29**.



Click *OK* to save the properties, then save your mod before continuing.

12) Set Phase Conditions

Updated script compatibility

Use PlagueRadioShuffler consistently for the quest attachment and every fragment kmyQuest selection. Select the conditional variable from that same quest script, and recompile the scene-begin and phase fragments against it. Track IDs remain zero-based, from 0 through iTrackCount - 1. The scene wiring below is retained from the original guide and has not been game-tested with this source. Verify in-game that one Randomize call corresponds to each track selection and that all audio entries play as intended.

Set Phase Conditions

This is the final step needed to connect everything together.

Each song needs a condition that tells the scene when it should play:

iCurrentTrack == (song index)

When the shuffle script changes *iCurrentTrack*, the matching song phase will run.

Double-click *Phase 1* above your first action.

Select the tab labeled *Start*.

In the *Conditions* area, right-click and choose:

New

Filter for:

GetVMQuestVariable

Select it from the dropdown.

Click:

Quest: 'YourQuest', NONE

Set the following:

- Quest: *MyRadioQuest*
- Parameter 2: *iCurrentTrack_var*

Click *OK*.

Now set:

- Comparison: ==
- Value: The track index number

The first track should use:

0

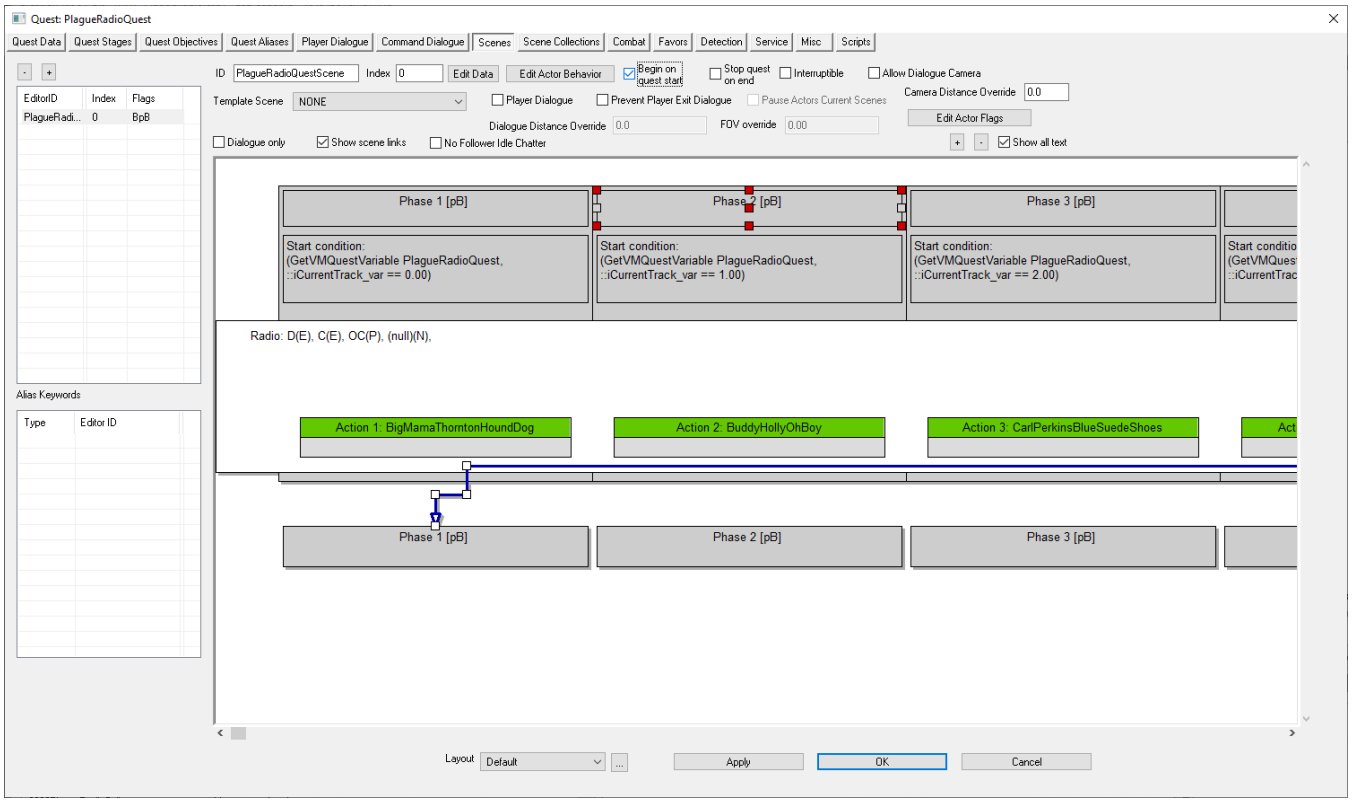
The second track should use:

1

The third track should use:

2

Continue this pattern for each song. Remember, the first track starts at zero.



Below the condition you just created, find the *Run Fragment* radio button.

Enable it.

In the dropdown next to the *kmyQuest* label, select PlagueRadioShuffler, matching the script attached to your quest.

In the text area, enter:

```
kmyQuest.Randomize()
```

Click *Compile*.

There should be no errors. If compilation fails, verify that your script name matches the name you created earlier and that the function call is typed correctly.

If everything compiles successfully, click *OK*.

Click *OK* again to return to the quest form.

This is a good time to close out of the quest form and save your mod.

Now repeat this process for every song phase.

For each phase:

- Set the condition to:

```
iCurrentTrack == [track index number]
```

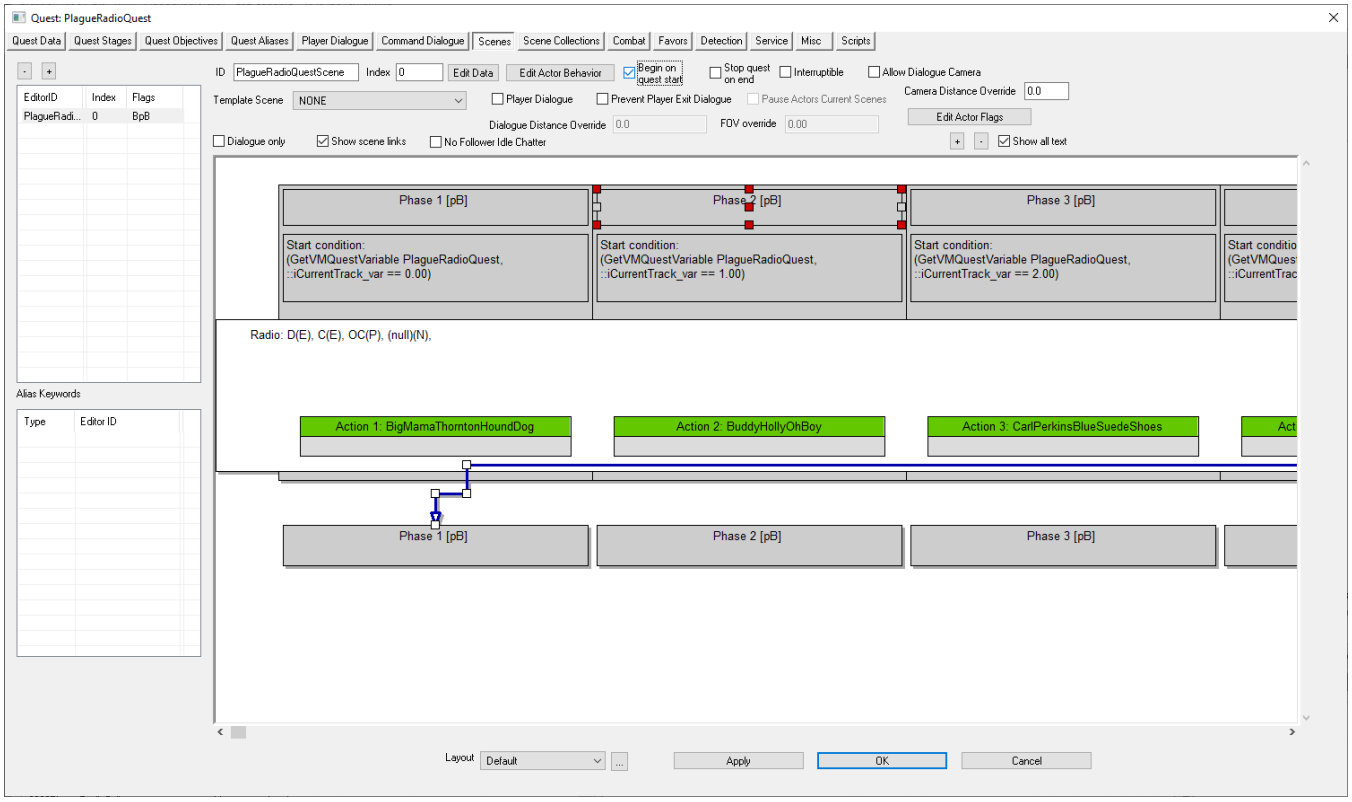
- Add the *Run Fragment* call:

```
kmyQuest.Randomize()
```

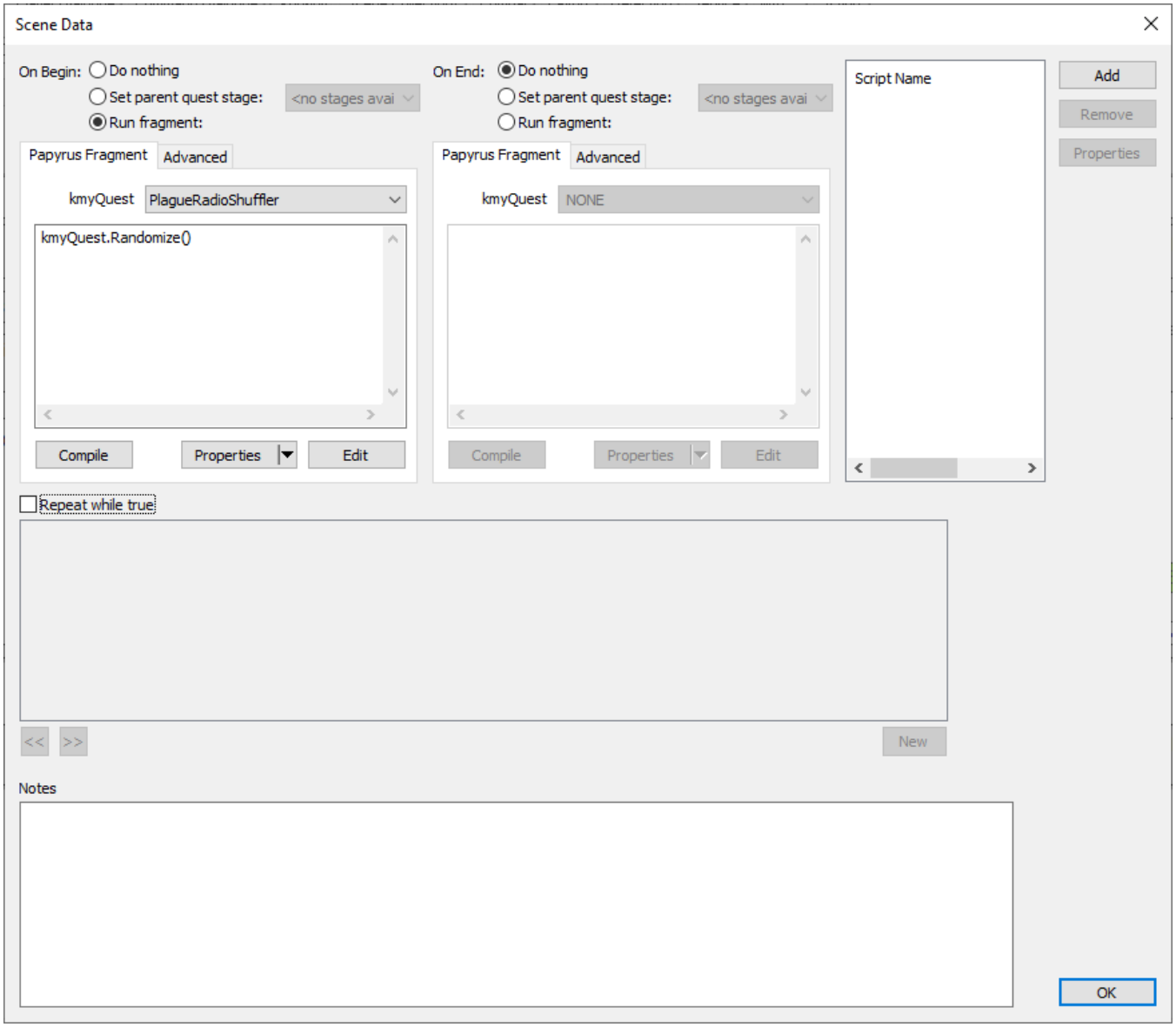
- Compile the fragment.

Remember that the track index starts at zero, so the condition for each phase should be one number lower than the song's position in your playlist.

Now make sure that "Begin on Quest Start" is checked, then click "Edit Data".



On the left of the resulting window under "On Begin" choose "Run Fragment" set "kmyQuest" to the shuffle script an enter "kmyQuest.Randomize() into the box below. Click "Compile" and you're done.



Save your mod when you're finished.

Ready to test?

You can continue to the optional receiver step, or load the game and test your station now.

Don't forget to enable your mod in your load order.

If everything was set up correctly:

- Your custom station should appear in the Pip-Boy Radio menu.
- Your station name should display correctly.
- Your music should begin playing when you "*Tune to*" the station.

  The most common reason a custom radio station does not play audio is incorrect compression settings.

If your radio station appears but no music plays, save yourself some troubleshooting time and return to the *Converting Songs* section.

Another common issue is file location

Double-check that your audio settings match exactly:

- Format: **.xwm**
- Location:
  Data\Sound\fx\mus\radio[YourModName]

Make sure the final files in that folder are your converted **.xwm** files.

Assuming everything went smoothly, congratulations. You now have your own custom Fallout 4 radio station.

13) *Optional: Make a Custom Radio, Jukebox, or Amplifier receiver.

Make a Custom Radio, Jukebox, or Amplifier receiver

As an optional step, you can create a custom receiver for your station. This can be a radio, jukebox, or amplifier.

This is probably the easiest part of the entire process, so why not?

In the *Object Window*, filter for:

*radio*receiver*

In the left pane, select:

WorldObjects > Activator

I chose:

RadioDiamondCityReceiver

Double-click it to open the form.

Change the Editor ID to something unique, for example:

"*MyRadioReceiver*"

Under *Radio Receiver Data*, set:

[]Enable *Is a Radio*[]Frequency: **92.55** (or whatever frequency you assigned to your transmitter)

Leave the remaining settings unchanged.

Click *OK*, then choose *Yes* when prompted to create a new form.

The same process can be used to create a custom jukebox or amplifier receiver. Simply start with the appropriate existing object instead of radio receiver.

To place your receiver in the world:

[]In the *Cell View* window, navigate to the location where you want the receiver to appear.[]Load that location.[]Drag and drop your new receiver into the Render Window.[]Adjust the position and rotation as needed.

Double-click the placed receiver and make sure:

Start Enabled

is checked.

The radio should appear to light up.

Test it in-game by traveling to the location where you placed the receiver and listening.

Keep in mind that radios and amplifiers are not as loud as the Pip-Boy radio, so increase the volume if you don't hear anything at. For max in world volume, use a jukebox (though still nto as loud as the pip-boy).

If you followed the guide, everything should now be working.

Congratulations, you've created your own custom Fallout 4 radio station!

* Appendix A: Station Content Tips

13) Station Content Tips

With the technical work out of the way, it's time for the fun part: giving your station a personality.

When I built my station, I wanted it to feel like a real Commonwealth radio station, not just a playlist with a name attached to it. That meant creating a DJ with a distinct personality.

After experimenting with a couple of different concepts, I settled on *Tal Conroy*.

With more than 2,200 hours wandering the Commonwealth, I had a pretty good feel for the world I wanted her to inhabit. Tal isn't a comedian or a news reporter. She's observant, quietly optimistic, and treats the strange things she encounters as just another part of life in the Wasteland.

I found it helpful to divide my station into several recurring segment types.

Commonwealth Commentary

These are short observations about the world, its people, and the odd little details that make Fallout 4 memorable - in-spite of its quirks.

Example:

> "I don't know who keeps posing teddy bears in these strange scenes all across the Commonwealth... but whoever you are, you have a very odd sense of humor."

Sponsor Messages

Fallout's prewar brands are perfect for tongue-in-cheek ads.

Example:

> "Still drinkin' that Slocum's Joe? Even after the bombs fell, it's still the finest brew in the wastes. Now with 40 percent less rads."

Stories From the Commonwealth

These are my favorite.

They're presented as rumors, caravan stories, or tales passed between travelers. Some may be true. Others probably aren't. That's part of the charm.

Example:

> "A traveler passing through said he once activated a Protectron in an old civic building."
>
> "Standard procedure. Nothing unusual."
>
> "Except the robot didn't stop following him when the job was done."
>
> "It just... kept walking."
>
> "No orders. No context. Just a very determined machine maintaining what it believed was companionship."
>
> "He tried everything. Running. Hiding. Even walking in circles until he got dizzy."
>
> "It always found him again."
>
> "Eventually he stopped trying to escape it."
>
> "Said it was quieter than most people he'd traveled with."
>
> "They parted ways near Goodneighbor."
>
> "The Protectron kept walking south. Still following instructions no one remembers giving it."

Breaking your station into recurring segment types like these makes it feel much more like a real radio station. It also makes writing new content easier, because you're not starting from a blank page every time. Instead, you're adding another story, another sponsor, or another observation from life in the Commonwealth.

Creating Voice Audio Files

You have some options here:

- Record audio using your own voice.
- Hire/trade/extort/manipulate/ask nicely (showbiz is brutal) someone else to do the voice acting.
- Call random people and ask them to speak the lines for you.
- Use resources that already exist in the game or the modding community.
- Use AI generated voice.

Not all of us are voice actors, and not everyone has the budget to hire one.

I decided on AI text-to-speech to create my DJ dialogue, station IDs, and other spoken content.

The best resource I've found for that so far is:

MiniMax Text-to-Speech[www.minimax.io]

Here's a sample of what you can do with it: https://www.youtube.com/watch?v=nZuX-UWSGN0

At the time of writing, it offers 10,000 free credits to new users who sell out their (disposable) email address (hint, hint), which is more than enough to generate hundreds of short radio segments. Just expect to regenerate lines a few times to get the pacing and inflection you're after.

AI voices have come a long way, but they're still far from perfect.

Sometimes the delivery is spot on.

Other times you'll get three different accents in a six-second clip, or the voice will decide halfway through the sentence that it's suddenly a completely different person. I've even had one clip sound like a man and a woman taking turns finishing each other's sentences, like some long-lost, slightly cursed *The Electric Company* skit.

The key is to keep your scripts relatively short. If a line doesn't sound right, regenerate it. You'll usually get a noticeably different read each time.

Don't waste your time or credits with the overhyped "Emotion" or "Sound Tag" features.

In my experience, you'll get much better results by using punctuation, sentence structure, and even the occasional phonetic spelling than by expecting the AI to understand or convincingly perform emotion.

The sound tags almost never fit naturally into a script. The laughter sounds more like a sneeze, the sighs are so exaggerated they become comical, and before you know it you've spent extra credits making the line sound worse instead of better.

With a little patience, and some clever use of commas, ellipses, question marks, and phonetic spellings, you can produce surprisingly natural-sounding dialogue that fits seamlessly between your music tracks.

So really, all that's left is writing the dialogue itself.

Sorry, but AI can't really help you there.

It might come up with a decent one-liner now and then, but when it comes to storytelling or developing a personality for your DJ, LLMs still have a habit of wandering off the rails, repeating themselves, or writing dialogue that sounds... well... like AI.

That's the part where your creativity comes in.

You're on your own there.

* Appendix B: Avoiding Copyright Problems

Avoiding Copyright Problems

If you're creating a radio station for your own personal use, use whatever music you like.

BUt - if you intend to distribute your mod, it's worth taking the time to document where your music came from. It can save you a lot of headaches later if someone questions the source of your audio.

The most reliable way to avoid issues is to stick to public domain compositions and obtain recordings from reputable public archives such as the Internet Archive or Library of COngress. Keep a simple text file with your project that lists the source of every recording you include.

It doesn't have to be complicated. Something like this is more than enough:

```
My Radio Station
Music Credits
=============

All music included with this sample project is believed to
be in the public domain in the United States.

The recordings were selected to demonstrate the mechanics
of a Fallout 4 custom radio station. They are not intended
to represent a complete soundtrack.

---

Maple Leaf Rag

Composer:
Scott Joplin (1868-1917)

Composition:
Published 1899. Public Domain.

Suggested Recording Source:
Internet Archive
https://archive.org/

Reference:
Library of Congress
https://www.loc.gov/collections/ragtime/

---

Solace (A Mexican Serenade)

Composer:
Scott Joplin (1868-1917)

Composition:
Published 1909. Public Domain.

Suggested Recording Source:
Internet Archive
https://archive.org/

Reference:
Library of Congress Ragtime Collection
https://www.loc.gov/collections/ragtime/

---

St. Louis Blues
```

```
Composer:
W. C. Handy (1873-1958)

Composition:
Published 1914. Public Domain.

Reference Recording:
Original Dixieland Jazz Band
Victor, 1921

Reference:
Library of Congress National Jukebox
https://www.loc.gov/item/jukebox-41556/

Suggested Download Source:
Internet Archive
https://archive.org/

---

Hard Times Come Again No More

Composer:
Stephen Foster (1826-1864)

Composition:
Published 1854. Public Domain.

Reference Recording:
Brunswick Quartette (1910)

Reference:
Library of Congress National Jukebox
https://www.loc.gov/item/jukebox-645406/

Suggested Download Source:
Internet Archive
https://archive.org/

---

Careless Love

Traditional American Folk Song

Composition:
Public Domain.

Suggested Recording Source:
Internet Archive
https://archive.org/

=========================================================
```

Will this absolutely guarantee you'll never receive a copyright claim?

No.

Copyright claims are usually automated based on a file signature, and they're frquently (notoriously) inaccurate.

What it does give you is a ***documented paper trail*** showing that you made a good faith effort to use public domain material and that you can identify the source of every recording in your mod.

That's a much better position to be in than trying to remember where you downloaded a song six months after releasing your mod.