

# DeepVoice AI Offline

Local On-Device Text-to-Speech for Unity



The editor window, the inference pipeline and the scripting API, explained from first launch to custom integration.

October 2026

# Contents

Chapters 1–7 cover using the window and the scene panel; 8–9 the preview player and audio tools; 10–11 workflows and settings; 12–15 internals and reference.

|                                                    |           |
|----------------------------------------------------|-----------|
| <b>01 Overview</b>                                 | <b>5</b>  |
| What is DeepVoice AI Offline? . . . . .            | 5         |
| How it works at a glance . . . . .                 | 5         |
| What is in the package . . . . .                   | 6         |
| Conventions in this guide . . . . .                | 6         |
| <b>02 Requirements and Installation</b>            | <b>7</b>  |
| Requirements . . . . .                             | 7         |
| Model folder layout . . . . .                      | 7         |
| Installation . . . . .                             | 8         |
| <b>03 Quick Start</b>                              | <b>9</b>  |
| What you get . . . . .                             | 9         |
| <b>04 The Scene Panel</b>                          | <b>10</b> |
| 4.1 What the scene panel is . . . . .              | 10        |
| 4.2 The panel at a glance . . . . .                | 10        |
| 4.3 Generating speech from the panel . . . . .     | 11        |
| 4.4 The settings, one by one . . . . .             | 12        |
| 4.5 Scene panel or Editor window? . . . . .        | 13        |
| 4.6 Remembered settings and reset . . . . .        | 13        |
| <b>05 The Generate Section</b>                     | <b>14</b> |
| 5.1 Window layout and behaviour . . . . .          | 14        |
| 5.2 The Generate section at a glance . . . . .     | 15        |
| 5.3 Text . . . . .                                 | 16        |
| 5.4 Voice Model Settings . . . . .                 | 16        |
| 5.5 Voice Library . . . . .                        | 16        |
| 5.6 Output . . . . .                               | 17        |
| 5.7 The Generate button and its blockers . . . . . | 18        |
| 5.8 What happens when you press Generate . . . . . | 18        |
| 5.9 Diagnostics . . . . .                          | 19        |
| <b>06 Generation Parameters</b>                    | <b>20</b> |

|                                                        |           |
|--------------------------------------------------------|-----------|
| How long is a frame? . . . . .                         | 20        |
| Reproducing a take . . . . .                           | 21        |
| Starting points to experiment from . . . . .           | 21        |
| <b>07 Voices and Cloning</b>                           | <b>22</b> |
| What a voice profile is . . . . .                      | 22        |
| Cloning a voice . . . . .                              | 22        |
| Getting a good clone . . . . .                         | 23        |
| Managing profiles . . . . .                            | 24        |
| <b>08 Preview Player</b>                               | <b>25</b> |
| How clips arrive in Preview . . . . .                  | 25        |
| <b>09 Audio Tools</b>                                  | <b>26</b> |
| 9.1 Audio Trimmer . . . . .                            | 26        |
| 9.2 Audio Joiner . . . . .                             | 27        |
| 9.3 Audio Equalizer . . . . .                          | 28        |
| <b>10 Workflows and Recipes</b>                        | <b>30</b> |
| Record a single voice-over line . . . . .              | 30        |
| Build a dialogue scene from several speakers . . . . . | 30        |
| Get a take back later . . . . .                        | 30        |
| Make a new character voice . . . . .                   | 31        |
| Use the audio in your game . . . . .                   | 31        |
| <b>11 Settings, Persistence and Files</b>              | <b>32</b> |
| What the window remembers . . . . .                    | 32        |
| Resetting everything . . . . .                         | 33        |
| Files the tool writes . . . . .                        | 33        |
| Model lifecycle . . . . .                              | 33        |
| <b>12 How It Works</b>                                 | <b>34</b> |
| 12.1 Architecture . . . . .                            | 34        |
| 12.2 The generation pipeline . . . . .                 | 34        |
| 12.3 Prompt layout . . . . .                           | 35        |
| 12.4 Model geometry . . . . .                          | 36        |
| 12.5 Key/value caches . . . . .                        | 36        |
| 12.6 Tokenizer . . . . .                               | 37        |
| 12.7 Audio I/O and DSP . . . . .                       | 37        |

|                                          |           |
|------------------------------------------|-----------|
| 12.8 Threading . . . . .                 | 37        |
| 12.9 Editor playback . . . . .           | 37        |
| <b>13 Scripting API</b>                  | <b>38</b> |
| Generating speech . . . . .              | 38        |
| Cloning a voice from a script . . . . .  | 39        |
| Processing audio . . . . .               | 40        |
| Runtime types . . . . .                  | 40        |
| Editor-only helpers . . . . .            | 42        |
| <b>14 Troubleshooting</b>                | <b>43</b> |
| Setup and model . . . . .                | 43        |
| Voices . . . . .                         | 43        |
| Generating . . . . .                     | 44        |
| Playback and audio tools . . . . .       | 45        |
| <b>15 Limits and FAQ</b>                 | <b>46</b> |
| Limits at a glance . . . . .             | 46        |
| Frequently asked questions . . . . .     | 46        |
| <b>Appendix A: runtime_manifest.json</b> | <b>48</b> |
| <b>Appendix B: Defaults at a Glance</b>  | <b>49</b> |
| <b>Appendix C: Script Inventory</b>      | <b>50</b> |
| Runtime assembly (Scripts/) . . . . .    | 50        |
| Editor assembly (Editor/) . . . . .      | 50        |
| <b>Appendix D: Glossary</b>              | <b>52</b> |

# 01

## CHAPTER

# Overview

A fully local text-to-speech toolkit for the Unity Editor. Type a sentence, choose or clone a voice, and get a ready-to-use audio clip in your project.

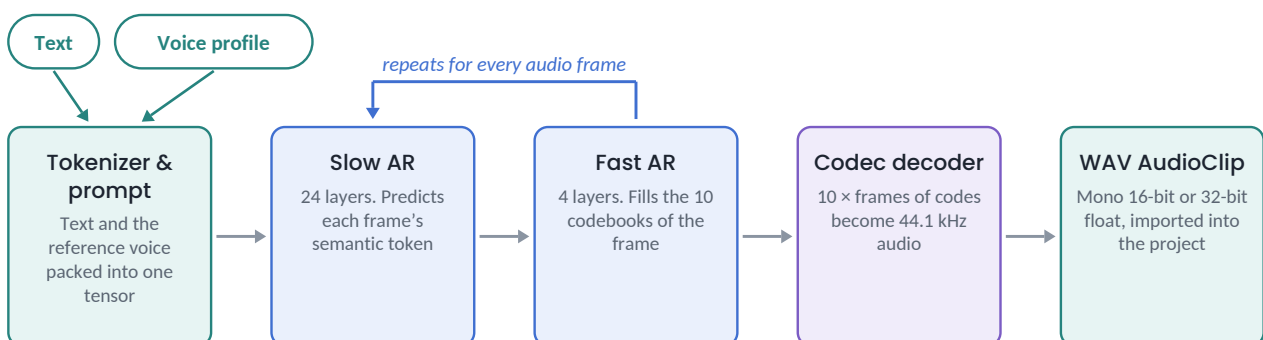
## What is DeepVoice AI Offline?

DeepVoice AI Offline is a Unity Editor tool and runtime library for **local, on-device text-to-speech**. It runs a neural voice model through ONNX Runtime on your own machine, so your text is turned into speech without being sent to a server. The scripts contain no network calls, and the model runs on the CPU.

The model is **zero-shot**: it does not ship with one fixed speaker. Instead it speaks in a *reference voice* that you pick from the Voice Library or clone from a short recording. Everything is driven from a single dockable window, **Window DeepVoice AI Offline**, and the results land in your project as ordinary `.wav` AudioClip assets that you can use like any other audio file.

|                                                                                                                            |                                                                                                                                                                         |                                                                                                                                         |
|----------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <b>Offline synthesis</b><br>Runs through ONNX Runtime on the CPU. Text and audio never leave your computer.                | <b>11 languages</b><br>English, Spanish, French, German, Italian, Dutch, Polish, Japanese, Korean, Chinese and Cantonese (EN, ES, FR, DE, IT, NL, PL, JA, KO, ZH, YUE). | <b>Voice library and cloning</b><br>Pick a voice with one click, or clone a new one from a 0.5–30 second recording plus its transcript. |
| <b>Tunable delivery</b><br>Control maximum length, output variation, creativity range, choice variety and the random seed. | <b>Preview and audio tools</b><br>Waveform player, trimmer, joiner and a five-band equalizer with volume and pitch, all inside the Editor.                              | <b>Scriptable runtime</b><br>A runtime assembly exposes generation, voice encoding and the audio DSP to your own scripts.               |

## How it works at a glance



Everything runs locally through ONNX Runtime on the CPU. Nothing is sent to a server.

**Figure 1.** From text to audio. The Slow AR network predicts one semantic token per audio frame; the Fast AR network then fills in that frame's ten codebooks; the codec decoder turns all frames into a waveform.

Your text and the selected voice's reference recording (stored as codec codes plus its transcript) are packed into a single prompt. Two small autoregressive transformers generate a sequence of audio *frames*, each about 46 ms long (2048 samples at 44.1 kHz), until the model emits its end marker or the length limit is reached. A codec decoder then converts the frames into a 44.1 kHz mono waveform, which is written to disk and imported by Unity. Chapter 12 explains each stage in more detail.

## What is in the package

The tool is split into two assemblies. The folders below are the defaults the window uses; you can change any of them from the window.

```
Assets/DeepVoice Offline/  
├ Editor/      # window and editor-only helpers (assembly ...Offline.Editor)  
├ Scripts/     # the runtime library           (assembly ...Offline.Runtime)  
├ Models/      # model files go here          (default "Model Folder")  
├ VoiceData/   # voice profile assets          (default "VoiceData Folder")  
└ Generated/   # audio the tool writes         (default "Output Folder")
```

The scripts this guide is based on are 25 C# files (16 editor, 9 runtime) and two assembly definitions. Appendix C lists every one and what it does. A compact scene panel (Chapter 4) offers basic generation on a single screen; its script is not part of that set.

## Conventions in this guide

- **Bold** text names something you see on screen, such as a button, field or menu item.
- **Monospace** text is a file name, folder, setting key or code identifier.
- Numbered markers in screenshots match the numbered legends printed beside them.
- Boxes flag extra context: **Note**, **Tip**, **Watch out** and **How it works**.

### NOTE

Values quoted in this guide, such as defaults, limits and ranges, were taken directly from the shipped scripts. Where a number is derived (for example seconds per frame) the working is shown. Values for the scene panel (Chapter 4) come from its screenshot, because its script is not among those scripts.

# 02

## CHAPTER

# Requirements and Installation

What you need before you generate your first clip, and how the model folder must be laid out.

## Requirements

| Component           | Requirement                                                                     | Notes                                                                                                          |
|---------------------|---------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| Unity Editor        | A version supported by the ONNX Runtime plugin                                  | The scripts use C# 8 syntax. Editor playback adapts to the audio API names used before and after Unity 2020.2. |
| ONNX Runtime plugin | <a href="https://github.com/asus4/onnxruntime">com.github.asus4.onnxruntime</a> | Referenced by both assemblies. Install it before importing or compiling the tool.                              |
| Model files         | The DeepVoice_Local model folder                                                | See “Model folder layout” below.                                                                               |
| Processor           | Any CPU                                                                         | Inference is CPU-only. More threads generate faster, up to the number of physical cores.                       |
| Memory              | About 1 GB while the model is loaded                                            | <b>Unload Model</b> frees roughly 1 GB. Cloning temporarily loads a further ~370 MB voice encoder.             |
| Disk (output)       | About 5.3 MB per minute of audio                                                | Mono 44.1 kHz. 16-bit PCM is the default; 32-bit float doubles that to about 10.6 MB per minute.               |

## Model folder layout

The model folder is read by the runtime and is deliberately *not* hard-wired to a project location: you choose it with the **Model Folder** field. The default is `Assets/DeepVoice Offline/Models`. It must contain a `runtime_manifest.json`, which tells the tool which graph files to load. (The **Output Folder** and **VoiceData Folder**, by contrast, must be inside `Assets` so Unity can import what the tool writes.)

```
Models/
├─ runtime_manifest.json           # geometry, token ids, rate, file precision
├─ slow_ar_int4.onnx (+ .onnx.data)
├─ fast_ar_int4.onnx (+ .onnx.data)
├─ codec_decoder_fp16.onnx
├─ tokenizer/
│   └─ tokenizer.json             # byte-level BPE vocabulary (about 12 MB)
└─ registration/
    └─ codec_encoder_fp16.onnx    # only needed for “Clone a Voice”
```

| File                               | Needed for               | Purpose                                                                                                               |
|------------------------------------|--------------------------|-----------------------------------------------------------------------------------------------------------------------|
| slow_ar_*.onnx /<br>fast_ar_*.onnx | Generate                 | The two autoregressive networks. Their weights live in <code>.onnx.data</code> files that must sit next to the graph. |
| codec_decoder_*.onnx               | Generate, voice previews | Turns codec codes into a 44.1 kHz waveform.                                                                           |
| tokenizer/tokenizer.json           | Generate                 | Converts text into token ids.                                                                                         |

| File                                 | Needed for    | Purpose                                                                  |
|--------------------------------------|---------------|--------------------------------------------------------------------------|
| registration/codec_encoder_fp16.onnx | Clone a Voice | Turns a reference recording into codec codes. Loaded only while cloning. |

#### HOW IT WORKS

File names are derived from the manifest: `slow_ar_{default_precision}.onnx` and `fast_ar_{default_precision}.onnx` (default precision `int4`), and the codec decoder named for `default_codec_precision` (default `fp16`). If your model uses different precision names, the manifest is what steers the loader.

## Installation

- 1 Import the package**

Import the DeepVoice Offline package into your project. Confirm `Assets/DeepVoice Offline` appears and that the console shows no compile errors.
- 2 Install the ONNX Runtime plugin**

Add `com.github.asus4.onnxruntime` to the project (for example through a scoped registry in `Packages/manifest.json`, shown below). The plugin's README at [github.com/asus4/onnxruntime-unity](https://github.com/asus4/onnxruntime-unity) lists the supported Unity versions, platforms and the current package version.
- 3 Add the model files**

Copy the model folder contents into `Assets/DeepVoice Offline/Models`, or keep them elsewhere and point the **Model Folder** field at them.
- 4 Open the window**

Choose **Window DeepVoice AI Offline**. Dock it next to the Inspector if you like; it is designed to stay usable when narrow.
- 5 Verify the model**

Expand **Diagnostics** and press **Inspect model files**. A healthy install ends with "OK - graph names, output counts and logits width match the port's expectations."

#### PACKAGES/MANIFEST.JSON (EXCERPT)

```
"scopedRegistries": [
  {
    "name": "NPM",
    "url": "https://registry.npmjs.com",
    "scopes": [ "com.github.asus4" ]
  }
],
"dependencies": {
  "com.github.asus4.onnxruntime": "<version from the plugin README>"
}
```

# 03

## CHAPTER

# Quick Start

From a fresh install to your first generated clip in six steps.

### 1 Open the window

**Window DeepVoice AI Offline.** The header shows **Model idle**: the model is not loaded yet, and that is normal.

### 2 Point to the model

In **Voice Model Settings**, make sure **Model Folder** points to the folder that contains `runtime_manifest.json`. A red message appears if it cannot be found.

### 3 Pick a voice

Expand **Voice** and click a voice button. The selected voice is highlighted and its sample plays in the Preview section. No voices yet? Clone one first (Chapter 7).

### 4 Type your text

Enter up to 2000 characters in the **Text** box. The counter underneath shows how much room is left.

### 5 Press Generate

A progress bar shows the frames produced so far, with a **Cancel** button. The first generation also loads the model, which takes a little longer; later ones reuse it.

### 6 Listen and refine

When finished, the clip is saved, selected in the Project window and loaded into **Preview**. Use the Audio Tools to trim, equalize or join it.

## What you get

A successful run saves a mono 44.1 kHz WAV into the Output Folder (default `Assets/DeepVoice Offline/Generated`) and shows a summary in the status box at the bottom of the window, in this form:

#### STATUS MESSAGE AFTER GENERATION

```
Created Assets/DeepVoice Offline/Generated/GeneratedVoice_01.wav
<length>s from <N> frames (<P> prompt positions, seed <S>) in <T>s.
```

The file name then advances automatically (`GeneratedVoice_01` becomes `GeneratedVoice_02`) so the next clip does not collide with it. The seed in the message lets you reproduce the take later (see Chapter 6).

#### TIP

Leave the Generate section open and keep the window docked. After each generation the new clip is already loaded in Preview, so you can listen, adjust the text and generate again without touching anything else.

#### NOTE

If **Generate** is greyed out, the red message beneath it says exactly why. Section 5.7 lists every possible reason.

# 04 WINDOW REFERENCE

## The Scene Panel

A compact, single-screen version of the generator: type, pick a voice, press Generate and save. It is the quickest way to hear what the model can do.

### 4.1 What the scene panel is

Besides the dockable Editor window described in Chapter 5, DeepVoice AI Offline can be shown as a compact panel in a scene. It is a one-screen interface for the same on-device text-to-speech engine, with the product name **DeepVoice Offline** and the tagline “On-device text to speech. No internet, no API keys.” in its header.

The panel is laid out in two columns. On the **left** you write the text and choose a voice, then press **Generate**. On the **right** are the **Preview** player, which plays and saves the result, and a collapsible **Settings** block with the generation parameters. A footer line points to the dockable window and an **Open window** button opens it.

### 4.2 The panel at a glance

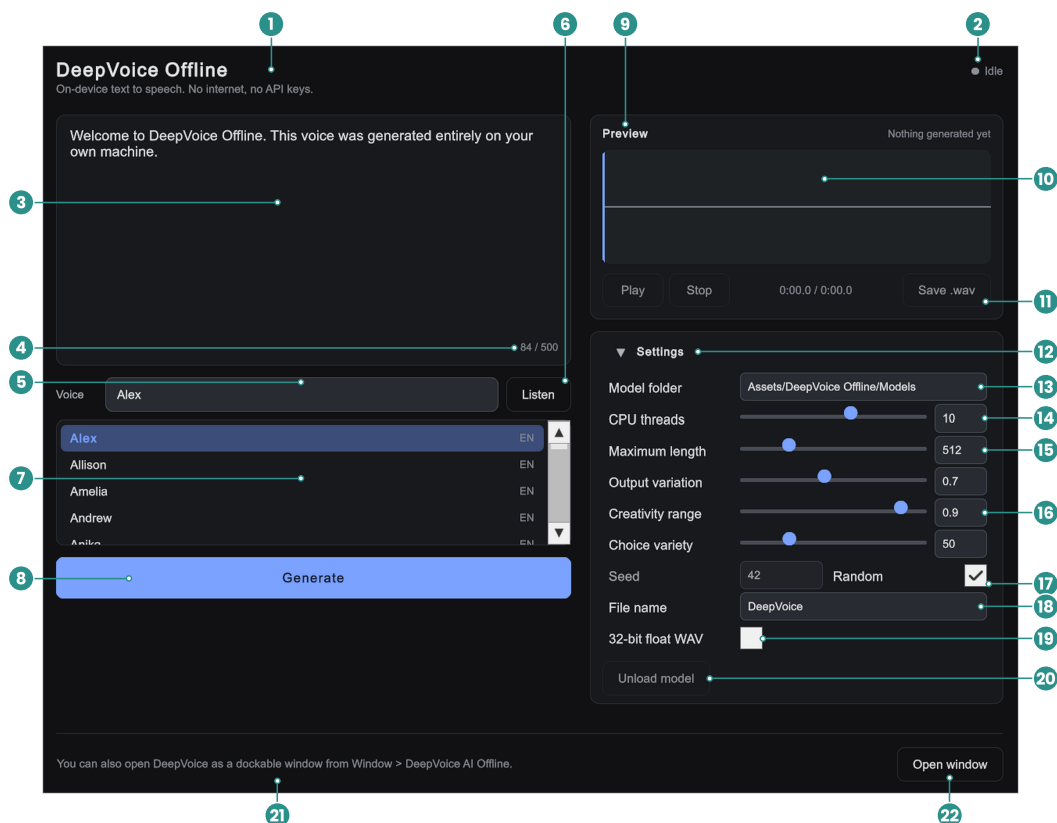


**Figure 2.** The scene panel on first launch: no clip generated yet, the voice Alex selected and the model not loaded (status Idle).

- 1 Title and tagline** – the product name, and the promise that nothing leaves your machine.
- 2 Status** – the state of the panel. It reads **Idle** when nothing is running.
- 12 Settings** – the collapsible block (▼) that holds everything below it.
- 13 Model folder** – where the model files are; shown with the default **Assets/DeepVoice Offline/Models**.

- 3 **Text box** – the text to speak. It is a multi-line field.
- 4 **Character counter** – characters used out of the limit: **84 / 500** here.
- 5 **Voice field** – the voice that is currently selected (*Alex*).
- 6 **Listen** – plays a sample of the selected voice before you generate.
- 7 **Voice list** – the available voices with language tags such as **EN**. The selected one is highlighted.
- 8 **Generate** – synthesizes the text with the selected voice and the settings on the right.
- 9 **Preview** – the player's header; *Nothing generated yet* until a clip exists.
- 10 **Waveform** – shows the generated clip. The blue bar at the left edge marks the start (**0:00.0**).
- 11 **Transport row** – **Play**, **Stop**, the time readout and **Save .wav** (writes the clip to a WAV file).
- 14 **CPU threads** – threads used for inference (10 in the screenshot).
- 15 **Maximum length** – the cap on clip length, in codec frames (512).
- 16 **Sampling sliders** – **Output variation**, **Creativity range** and **Choice variety** (0.7, 0.9, 50).
- 17 **Seed and Random** – the seed (42) is greyed out while **Random** is ticked.
- 18 **File name** – base name for saved audio (*DeepVoice*).
- 19 **32-bit float WAV** – save 32-bit float instead of 16-bit audio. Unticked.
- 20 **Unload model** – releases the model from memory. Greyed out while none is loaded.
- 21 **Hint line** – points to the dockable window: **Window DeepVoice AI Offline**.
- 22 **Open window** – opens that dockable window.

**NOTE**

**How this chapter was written.** The scripts this guide is based on contain the Editor window and the runtime library; the scene panel's own script is not among them. This chapter therefore rests on the screenshot, on the labels of the controls, on the Editor window's matching controls, and on the setting names the Editor window still clears for this panel (section 4.6).

The Editor window's code refers to the panel as “an earlier scene UI”. If the panel in your installed version looks different from the screenshot, trust what you see on screen.

## 4.3 Generating speech from the panel

- 1 **Write the text**  
Type or paste what the voice should say. The counter at the lower right of the box shows how much of the **500-character** limit you have used.
- 2 **Choose a voice**  
Click a voice in the list. It is highlighted and its name appears in the field above. Press **Listen** to audition it first.
- 3 **Check the settings**  
The defaults suit most speech. Make sure **Model folder** points at the folder that contains `runtime_manifest.json` (Chapter 2). Chapter 6 explains the sliders.
- 4 **Press Generate**  
The first generation also loads the model, so it takes a little longer. Afterwards the model stays in memory until you press **Unload model**.
- 5 **Listen and save**  
The result appears in **Preview**. Use **Play** and **Stop** to listen, and **Save .wav** to keep it.

**TIP**

Untick **Random** to use the same seed every time. The same text, voice and settings then give the same take, which is handy when you are comparing two settings. Chapter 6 explains how seeds work.

## 4.4 The settings, one by one

Every control here has a counterpart in the Editor window, and the engine behind both is the same, so each setting has the effect described in Chapter 6. The values in the screenshot are, apart from the thread count and file name, the Editor window's own defaults.

| Control                 | In the screenshot                  | What it does                                                                                                                                                                                                                            |
|-------------------------|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Model folder</b>     | Assets/DeepVoice<br>Offline/Models | The folder holding <code>runtime_manifest.json</code> and the model files (Chapter 2).                                                                                                                                                  |
| <b>CPU threads</b>      | 10                                 | How many CPU threads inference may use. More threads are faster, up to the number of physical cores. The Editor window starts at 4. As there, the most reliable way to apply a changed value is to unload the model and generate again. |
| <b>Maximum length</b>   | 512                                | A safety cap on clip length in codec frames of about 46.4 ms each, so 512 frames is up to <b>23.8 s</b> . Generation normally stops by itself when the speech ends.                                                                     |
| <b>Output variation</b> | 0.7                                | How much delivery differs from take to take. Lower is steadier, higher is more expressive but less reliable (this is the temperature).                                                                                                  |
| <b>Creativity range</b> | 0.9                                | How far from the safest choice the voice may wander (top-p).                                                                                                                                                                            |
| <b>Choice variety</b>   | 50                                 | How many alternatives are weighed at every step (top-k).                                                                                                                                                                                |
| <b>Seed</b>             | 42                                 | The starting value of the random generator. Greyed out while <b>Random</b> is on.                                                                                                                                                       |
| <b>Random</b>           | ticked                             | Draws a new seed for every generation, so pressing Generate again gives a new take.                                                                                                                                                     |
| <b>File name</b>        | DeepVoice                          | The base name used when audio is saved. (The Editor window's default is <code>GeneratedVoice_01.</code> )                                                                                                                               |
| <b>32-bit float WAV</b> | unticked                           | Saves 32-bit floating-point samples instead of 16-bit. It avoids clipping and doubles the file size.                                                                                                                                    |
| <b>Unload model</b>     | greyed out                         | Releases the loaded model from memory. It is greyed out while no model is loaded.                                                                                                                                                       |

## 4.5 Scene panel or Editor window?

The two are front ends for the same runtime. Use the scene panel for quick generation; use the Editor window for everything else. The **Open window** button in the footer takes you straight there.

|                              | Scene panel                                                     | Editor window                                                         |
|------------------------------|-----------------------------------------------------------------|-----------------------------------------------------------------------|
| <b>Text limit</b>            | 500 characters (counter reads 84 / 500)                         | 2000 characters                                                       |
| <b>Choosing a voice</b>      | A scrolling list with language tags, and a <b>Listen</b> button | The Voice Library grid; clicking a voice plays its sample (Chapter 5) |
| <b>Cloning a voice</b>       | Not shown                                                       | <b>Clone a Voice</b> panel (Chapter 7)                                |
| <b>Result</b>                | <b>Preview</b> player with a <b>Save .wav</b> button            | Written to the Output Folder as an AudioClip and loaded into Preview  |
| <b>Audio tools</b>           | Not shown                                                       | Trimmer, Joiner and Equalizer (Chapter 9)                             |
| <b>Diagnostics</b>           | <b>Unload model</b> only                                        | Load / Unload, Inspect model files, Reset Saved Settings              |
| <b>Folders</b>               | Model folder                                                    | Model, Output and VoiceData folders                                   |
| <b>Remembers settings in</b> | PlayerPrefs ( <code>DeepVoice.Scene.*</code> )                  | EditorPrefs and the window layout (Chapter 11)                        |

## 4.6 Remembered settings and reset

The Editor window's **Diagnostics Reset Saved Settings** (section 5.9) also deletes twelve Unity PlayerPrefs keys that all start with `DeepVoice.Scene..`. They are the settings this panel remembers, and their names line up with its controls one for one. Generated audio and voice profiles are never touched:

| Key       | Control        | Key        | Control          |
|-----------|----------------|------------|------------------|
| Text      | Text box       | Variation  | Output variation |
| Voice     | Selected voice | Creativity | Creativity range |
| Model     | Model folder   | Variety    | Choice variety   |
| Threads   | CPU threads    | Seed       | Seed             |
| MaxLength | Maximum length | RandomSeed | Random           |
| FileName  | File name      | FloatWav   | 32-bit float WAV |

# 05

WINDOW REFERENCE

## The Generate Section

A guided tour of the window and every control used to turn text into audio.

### 5.1 Window layout and behaviour

The window is built to be docked like the Inspector, so every control survives a narrow panel (minimum size 330 × 480 pixels). It has a header, three colour-coded collapsible sections and a footer.

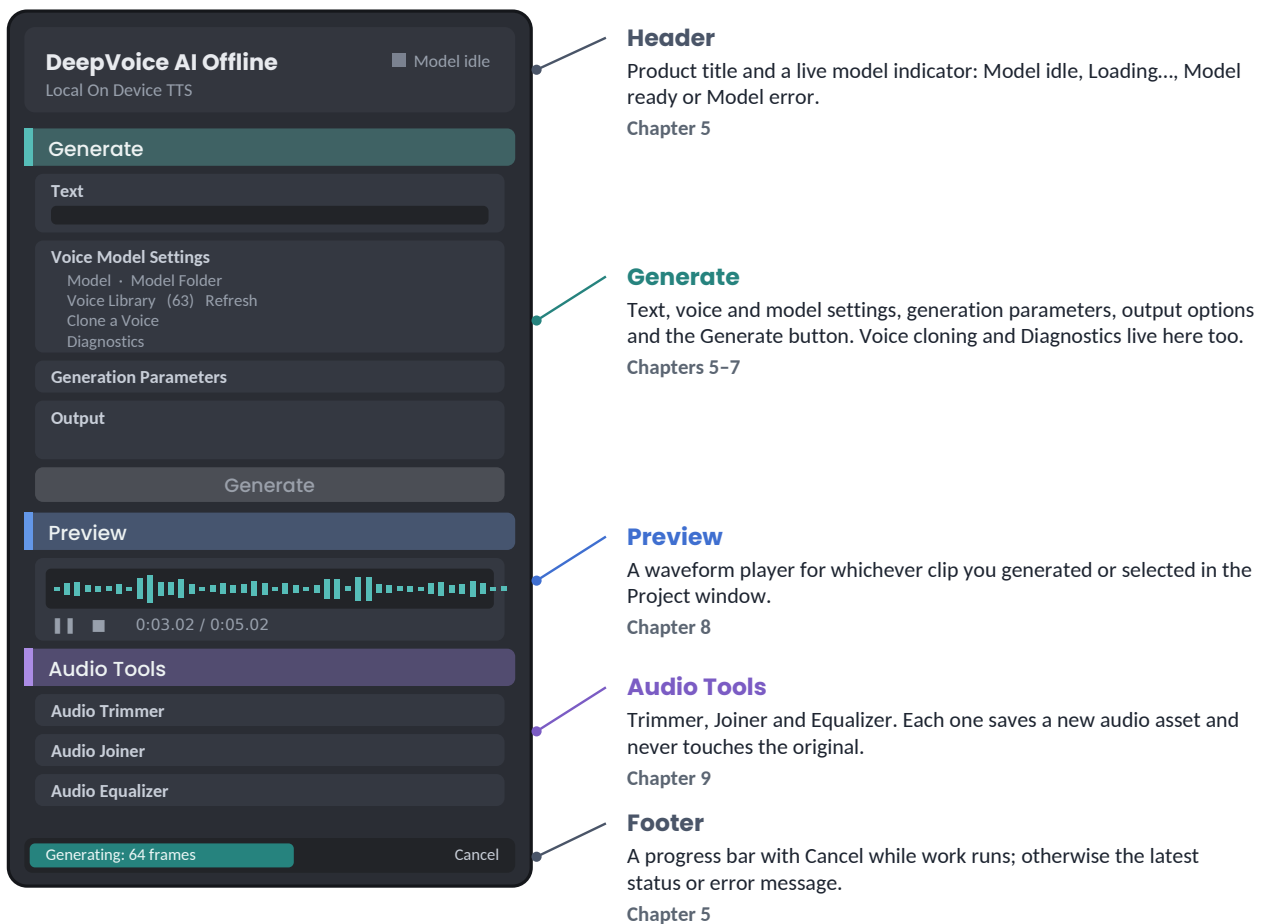


**Figure 3.** Anatomy of the DeepVoice AI Offline window. Section colours are used throughout this guide: teal for Generate, blue for Preview, violet for Audio Tools.

## The model indicator

The small square and label in the top-right corner always show the state of the loaded model:

| Indicator   | Colour | Meaning                                                                                     |
|-------------|--------|---------------------------------------------------------------------------------------------|
| Model idle  | Grey   | No model is loaded. It loads on the first generation, or when you press <b>Load Model</b> . |
| Loading...  | Amber  | The model files are being opened.                                                           |
| Model ready | Green  | The model is in memory and the next generation starts immediately.                          |
| Model error | Red    | Loading failed. The reason is shown in Diagnostics and in the footer.                       |

## While work is running

Generation, cloning, model loading and inspection run on a background thread. While one is running every section is disabled, the footer shows a progress bar with a **Cancel** button, and Unity is told not to reload assemblies, so a script recompile cannot pull the model out from under a running job. When the work ends the footer shows the outcome: an info message, or a red error if something failed. Cancelling shows “Cancelled.”

## 5.2 The Generate section at a glance

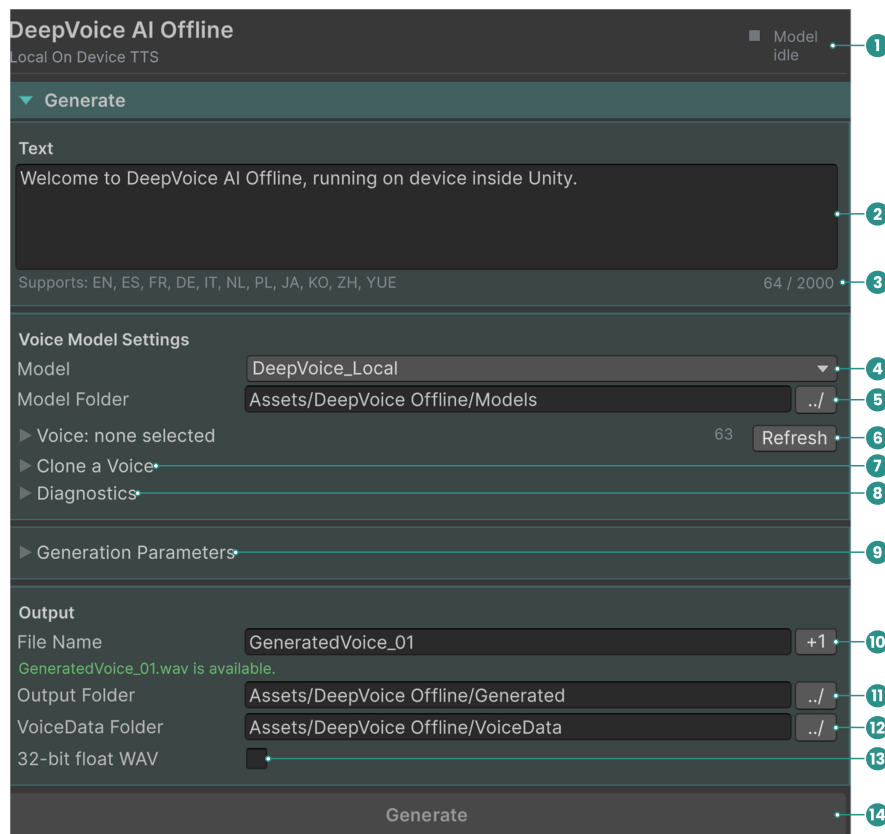


**Figure 4.** The Generate section as it appears on first launch with no voice selected. The Generate button stays grey until every requirement is met.

- 1 **Header** – title and live model indicator.
- 2 **Text** – what the voice will say (up to 2000 characters).
- 3 **Languages and counter** – supported languages and characters used.
- 4 **Model** – the voice model; currently DeepVoice\_Local.
- 5 **Model Folder** – where the model files live, with a folder picker.
- 6 **Voice Library** – foldout, voice count and a Refresh button.
- 7 **Clone a Voice** – register a new voice from a recording.
- 8 **Diagnostics** – threads, load/unload, inspection and reset.
- 9 **Generation Parameters** – length, variation, creativity, variety, seed.
- 10 **File Name** – output name, +1 button and availability message.
- 11 **Output Folder** – where generated clips are written.
- 12 **VoiceData Folder** – where voice profiles are stored and scanned.
- 13 **32-bit float WAV** – optional higher-precision output.
- 14 **Generate** – starts synthesis.

## 5.3 Text

Type or paste the text to speak. The field wraps like any text box and grows downwards as you add lines, between about 70 and 320 pixels tall. Input is limited to **2000 characters**; anything beyond that is cut off, the counter turns red and “Character limit reached.” appears.

Before synthesis the text is cleaned: control, format, private-use and unassigned characters are removed, runs of whitespace collapse into single spaces, and a line break that sits between two CJK characters is removed instead of becoming a space. The line under the box lists the supported languages: **EN, ES, FR, DE, IT, NL, PL, JA, KO, ZH and YUE**.

### TIP

Write the way you want it spoken. Punctuation guides pacing, and spelling out numbers, dates and abbreviations in words gives you control over how they are read. Very long passages are best generated in several shorter clips and joined with the Audio Joiner.

## 5.4 Voice Model Settings

### Model

A drop-down listing the available voice models. The package ships **DeepVoice\_Local**, which runs entirely on this machine’s CPU.

### Model Folder

The folder holding `runtime_manifest.json` and the model files. Paths inside the project are displayed from `Assets/` down, while the stored value is an absolute path so background threads can use it. Type a path or press `../` to browse. Changing the folder clears any previous model error. If the manifest is missing, a red message says so, and Generate stays disabled.

## 5.5 Voice Library

The **Voice** foldout lists every voice profile found in the VoiceData Folder. Its title shows the current selection (**Voice: none selected** in the screenshot above), the figure to its right is the number of voices found, and **Refresh** rescans. The library also refreshes by itself whenever the project contents change.

- **Selecting.** Click a voice button. The selected voice is tinted and shown in bold, and its sample starts playing in Preview. The grid wraps to the panel width from one to four columns.
- **Sample playback.** A voice registered in Unity plays the recording it was cloned from. A voice without a clip (for example one imported from another runtime) is previewed by decoding its codec codes through the codec decoder, which needs the Model Folder to be set. Decoded previews are cached for the session.
- **Hover tooltip.** Shows the voice name, reference length in seconds, language tag, number of codec frames and the start of the reference text.
- **Unusable voices.** A profile with no frames, a mismatched code array or an empty transcript is greyed out; its tooltip reads “Unusable:” followed by the reason.
- **Search scope.** The configured VoiceData Folder is searched first. If it holds no profiles, the whole project is searched, so voices still appear after being moved. The line under the grid says which scope was scanned.

**HOW IT WORKS**

If voice assets exist but do not appear, the library explains why and may offer a one-click **Repair voice profiles** button. See “Managing profiles” in Chapter 7.

## 5.6 Output

| Control                 | What it does                                                                                                                                                                                                                                                                                                        |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>File Name</b>        | Name of the <code>.wav</code> to write. Characters other than letters, digits, <code>_</code> <code>-</code> <code>.</code> and spaces become underscores. A green line says the name is available; a red line warns that it exists and <b>would be overwritten</b> . After every save a trailing number is bumped. |
| <b>+1</b>               | Jumps to the next unused <code>Name_NN</code> in the Output Folder.                                                                                                                                                                                                                                                 |
| <b>Output Folder</b>    | Where clips are written. Must be inside <code>Assets</code> ; otherwise “Folder must be inside Assets.” is shown. <code>../</code> opens a folder picker.                                                                                                                                                           |
| <b>VoiceData Folder</b> | Folder scanned for voice profiles and where new clones are saved. Picking a folder outside the project is refused.                                                                                                                                                                                                  |
| <b>32-bit float WAV</b> | Writes 32-bit floating-point samples instead of 16-bit PCM. It avoids clipping if the codec output exceeds full scale, at twice the file size. The setting also applies to the Trimmer, Joiner and Equalizer.                                                                                                       |

**WATCH OUT**

Generate overwrites a file that already has the chosen name (the red warning tells you first). The Trimmer, Joiner and Equalizer never overwrite: they pick a unique name instead.

## 5.7 The Generate button and its blockers

Generate turns teal when everything is ready. Otherwise it stays grey and a red message beneath it names the first unmet requirement. The checks run in this order:

| Message                                                  | What to do                                                   |
|----------------------------------------------------------|--------------------------------------------------------------|
| Enter some text to synthesize.                           | Type something in the Text box.                              |
| Set the model folder.                                    | Fill in the Model Folder field.                              |
| runtime_manifest.json was not found in the model folder. | Point Model Folder at the folder that contains the manifest. |
| Select a voice from the Voice Library.                   | Click a voice. This is the state shown in the screenshot.    |
| The selected voice is unusable: ...                      | Choose another voice or fix the profile (Chapter 7).         |
| Enter an output file name.                               | Give the clip a name.                                        |
| The output folder must be inside Assets.                 | Choose an Output Folder under <b>Assets</b> .                |

## 5.8 What happens when you press Generate

### 1 Snapshot

The selected profile is copied into plain data on the main thread, because Unity objects must not be touched from a worker thread. If **Random** is on, a fresh seed is drawn now.

### 2 Load model

The footer reads “Loading model...”. The model is loaded once and reused as long as the Model Folder does not change.

### 3 Build prompt

Your text and the voice’s reference are packed into the model prompt.

### 4 Generate frames

The label counts frames and the approximate seconds produced, for example “Generating: 120 frames (~5.6s)”. Press **Cancel** at any time.

### 5 Decode and save

The codec decoder renders the waveform, a WAV is written and imported, the new clip is selected in the Project window, loaded into Preview, and the file name is advanced.

### 6 Report

The status box shows the file, its length, frame count, prompt size, seed and elapsed time. If generation ended at the length cap it adds “Stopped at the frame cap”; raise **Maximum Length**.

## 5.9 Diagnostics

Expand **Diagnostics** (inside Voice Model Settings) for the tools you use when something is slow or wrong.

| Control                     | What it does                                                                                                                                                                                        |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>CPU threads</b>          | ONNX Runtime thread count, from 1 to 16 (default 4). More threads are faster up to the number of physical cores. The value is applied when the model is loaded.                                     |
| <b>Load Model</b>           | Loads the model now and reports the result, without generating audio. The status box then lists layers, codebooks, output sample rate and tokenizer size.                                           |
| <b>Unload Model</b>         | Releases the model and frees about 1 GB of RAM. The next generation reloads it.                                                                                                                     |
| <b>Inspect model files</b>  | Reads the real ONNX graphs and checks them against what the tool expects. The report is logged to the Console and saved as <code>Library/DeepVoice_Local/model_inspection.txt</code> .              |
| <b>Reset Saved Settings</b> | Forgets remembered folders, file name, selected voice and every other setting, then restores the defaults. Generated audio and voice profiles are not touched. A confirmation dialog appears first. |
| <b>Status line</b>          | Plain-language model state: <i>Loading...</i> , <i>Error loading model: ...</i> , <i>Model loaded successfully from ...</i> or <i>Model not loaded</i> .                                            |

### WATCH OUT

Changing **CPU threads** does not reload a model that is already in memory. To apply a new value, press **Unload Model** and then generate or press **Load Model** again.

# 06

CHAPTER

## Generation Parameters

What each slider does, how generation length is counted, and how to reproduce a take you like.

Open **Generation Parameters** in the Generate section to reach these controls. Defaults suit most speech; change one thing at a time so you can hear its effect.

| Control                 | Range          | Default | What it changes                                                                                                                                                                        |
|-------------------------|----------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Maximum Length</b>   | 32–2000 frames | 512     | A safety cap on clip length. Generation normally stops by itself when the speech finishes. Raise it if long text is cut off. The label under the slider shows the approximate seconds. |
| <b>Output Variation</b> | 0.05–1.5       | 0.7     | How much delivery differs from take to take. Low is steady and predictable; high is more expressive but less reliable.                                                                 |
| <b>Creativity Range</b> | 0.1–1.0        | 0.9     | How far from the safest choice the voice may wander. Lower sticks to the most natural option; higher allows unusual delivery.                                                          |
| <b>Choice Variety</b>   | 1–200          | 50      | How many alternatives are weighed at every step. Lower is more consistent and slightly flatter; higher gives more range.                                                               |
| <b>Seed</b>             | any integer    | 42      | Same text, voice, settings and seed give the identical take. Disabled while <b>Random</b> is on.                                                                                       |
| <b>Random</b>           | on / off       | on      | Draws a new seed for every generation so pressing Generate again rerolls the take.                                                                                                     |

### HOW IT WORKS

The labels are friendly names for standard sampling controls. **Output Variation** is temperature, **Creativity Range** is top-p (nucleus sampling), **Choice Variety** is top-k and **Maximum Length** is the maximum number of codec frames.

## How long is a frame?

Maximum Length is counted in codec frames, not seconds. One frame is 2048 samples at 44.1 kHz, which is **about 46.4 ms**, or roughly 21.5 frames per second. The slider's hint line uses exactly this conversion.

| Maximum Length | Approximate audio |
|----------------|-------------------|
| 64             | 3.0 s             |
| 128            | 5.9 s             |
| 256            | 11.9 s            |
| 512            | 23.8 s            |
| 1024           | 47.6 s            |
| 2000           | 92.9 s            |

**WATCH OUT**

Length is bounded by the model's context window of **2048 positions**, and the prompt uses some of it: the instruction text, your text and every frame of the reference recording. Frames available for new audio = 2048 – prompt length (never more than your Maximum Length). A long reference or long text therefore shortens the maximum clip. Chapter 7 shows how much a reference recording costs.

## Reproducing a take

### 1 Generate with Random on

Each press of Generate uses a new seed. Listen until you find a take you like.

### 2 Read the seed

The status box ends with `seed <number>` for the clip just created.

### 3 Turn Random off and enter the seed

Keep the same text, voice and parameters. The same inputs now reproduce that take.

**NOTE**

Seeds are reproducible **inside Unity**. They are drawn from a xoshiro256\*\* generator, which cannot match the random stream of the original Python reference runtime, so the same seed will not give the same audio there.

## Starting points to experiment from

These are not presets in the tool; they are suggestions built from what each control does. Change one value at a time and listen.

| Goal                         | Output Variation | Creativity Range | Choice Variety |
|------------------------------|------------------|------------------|----------------|
| Steady, consistent narration | 0.5              | 0.85             | 30             |
| Balanced default             | 0.7              | 0.9              | 50             |
| More expressive reads        | 0.9              | 0.95             | 80             |

### How a token is chosen

For readers who want the detail: at every step the model produces a score for each candidate token. The tool keeps only the top  $k$  candidates, trims that list to the smallest set whose probabilities add up to  $p$ , divides the scores by the temperature and then draws one with the Gumbel-max trick. For the semantic token there is an extra guard: a second candidate is drawn at temperature 1.0 with top-p 0.9, and is used instead whenever the first draw repeats one of the last ten semantic tokens. This discourages stuttering loops without changing your settings.

# 07

## CHAPTER

# Voices and Cloning

How voice profiles work, how to clone your own, and how to keep a voice library healthy.

## What a voice profile is

A voice is stored as a **DeepVoiceProfile** asset (a ScriptableObject) in your VoiceData folder. It is the Unity equivalent of the Python runtime's `voices/<name>/` folder and holds everything the model needs to imitate a speaker:

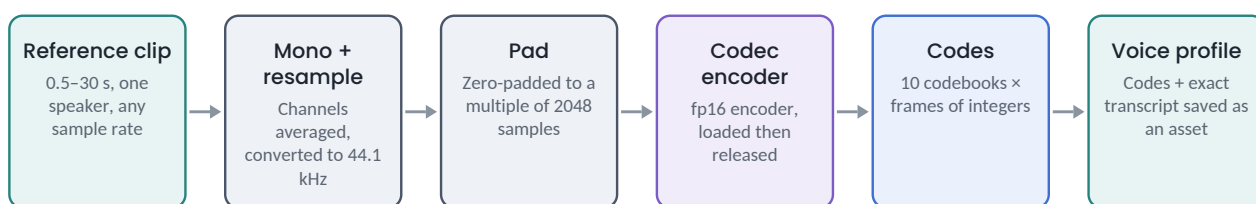
| Field                         | Meaning                                                                                                              |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------|
| voiceName                     | Display name shown in the Voice Library. Default “speaker”.                                                          |
| referenceText                 | The <b>exact transcript</b> of the reference recording. Accuracy here strongly affects quality.                      |
| numCodebooks / frames / codes | The reference encoded as a [numCodebooks × frames] matrix of codec codes (10 codebooks for this model), stored flat. |
| sourceSampleRate              | Sample rate of the original recording.                                                                               |
| modelFingerprint              | Fingerprint of the model the voice was registered with, recorded for reference.                                      |
| referenceClip                 | The original recording. Optional. Used for instant preview in the library.                                           |
| language                      | Free-form tag shown in the tooltip, for example “EN”.                                                                |

A profile is *usable* when it has at least one frame, the code array matches numCodebooks × frames, and the transcript is not empty.

## Cloning a voice

Open **Clone a Voice** inside Voice Model Settings. The cloning panel contains:

| Field            | Details                                                                                                            |
|------------------|--------------------------------------------------------------------------------------------------------------------|
| Reference clip   | An AudioClip of 0.5–30 seconds with a single speaker, as clean as possible. Any sample rate works.                 |
| Exact transcript | What is said, word for word. The tooltip notes this matters more for quality than any other setting.               |
| Voice name       | Used for the profile asset’s file name.                                                                            |
| Language tag     | A small field beside the name; defaults to “EN”.                                                                   |
| Clone Voice      | Starts cloning. Disabled until a clip, a transcript and a model folder are set; a hint line says which is missing. |



The Clone Voice button runs this pipeline on a background thread.

**Figure 5.** What the Clone Voice button does behind the scenes.

When cloning finishes, the new profile is saved as `<VoiceData folder>/<name>.asset` (a unique name is chosen if that exists), selected as the current voice, highlighted in the Project window and listed in the Voice Library. The status box reports, for example, “Cloned ‘name’ from a Xs reference.”

#### HOW IT WORKS

Cloning loads the voice encoder (about 370 MB) and releases it immediately afterwards, so it does not sit in memory beside the generation model. It needs `registration/codec_encoder_fp16.onnx` in the model folder.

## Getting a good clone

- **One speaker, clean audio.** Avoid music, background voices, heavy reverb and clipping.
- **Type the transcript exactly.** Every word as spoken. The transcript is part of the prompt, so errors teach the model the wrong alignment.
- **Keep it short.** A few seconds is often enough. The sample voices shown in this guide’s screenshots are roughly 5–6.5 seconds long.
- **Match the language.** Use a reference in the language you plan to synthesize and set the language tag so you can find it later.
- **Mind the context budget.** Every frame of the reference is part of the 2048-position prompt, so long references leave less room for generated audio.

The table shows what a reference recording costs. Each second adds about 21.5 frames to the prompt, and a reference is padded up to a multiple of 2048 samples before encoding.

| Reference length | Approx. frames in the prompt | Share of the 2048-position window |
|------------------|------------------------------|-----------------------------------|
| 5 s              | 108                          | 5%                                |
| 10 s             | 216                          | 11%                               |
| 15 s             | 323                          | 16%                               |
| 30 s             | 646                          | 32%                               |

The prompt also carries the instruction text, the reference transcript and your own text, so the real total is somewhat higher. If the prompt reaches the limit, generation stops with: *Prompt is N positions, which exceeds the model limit of 2048. Use a shorter reference recording or shorter text.*

## Errors you may meet while cloning

| Message                                           | Meaning                                                                             |
|---------------------------------------------------|-------------------------------------------------------------------------------------|
| Reference audio must be 0.5-30 seconds, got X s.  | The clip is outside the allowed range. Trim it with the Audio Trimmer first.        |
| codec_encoder_fp16.onnx not found...              | Add the <code>registration/</code> folder to the model folder.                      |
| Could not read 'clip' / Unity would not decode... | Use a <code>.wav</code> , or set the clip's import Load Type to Decompress On Load. |

## Managing profiles

### Moving, copying and sharing

Profiles are ordinary project assets. To use a voice in another project, copy its `.asset` file (and its reference clip, if you want the instant preview) into that project's VoiceData folder. A voice can only be used with a model that has the same number of codebooks; otherwise generation stops with *Voice 'x' has N codebooks but the model expects M.*

### Repairing voices that do not load

A profile stores a reference to its script by GUID. If the script is replaced by a file with a new GUID (for example after a class rename), Unity shows "Missing (Mono Script)" and the Voice Library comes up empty even though the data is intact. The library detects this and shows a warning with a **Repair voice profiles** button.

- The repair **relinks the script reference only**; it does not alter voice data.
- Before anything is changed, each file is copied to `Library/DeepVoice_Local/ProfileBackup/<timestamp>/`.
- A file is only treated as a profile (and only ever rewritten) if it is text-serialized and contains all five expected fields.
- Binary-serialized assets cannot be repaired this way. Set Unity's Asset Serialization to *Force Text* if you see "none are DeepVoice voice profiles that Unity can read as text."

### Importing voices from the Python runtime

The runtime includes `NpyReader.ReadCodes`, which reads the `.npy` code matrix written by the Python registration step (unsigned or signed 16-bit, 32-bit and 64-bit integer arrays in C order, shaped [codebooks, frames] or [1, codebooks, frames]). The window itself has no import button, so importing is done from a script: read the codes, create a profile, call `SetCodes`, set the transcript and save the asset. Profiles imported this way have no reference clip; the library previews them by decoding their codes.

# Preview Player

A waveform player built into the window. It plays any clip inside the Editor, without entering Play Mode.

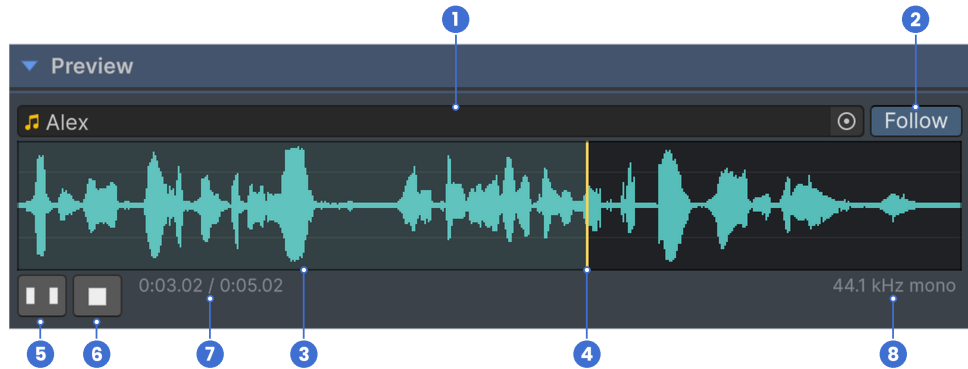


**Figure 6.** The Preview section with a clip loaded and playing. The yellow line is the playhead; the part already played is brightened.

- |                                                                                                                                                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>1 Clip field</b> – the clip being previewed. Drag an AudioClip here or use the picker.</p> <p><b>2 Follow</b> – when on, Preview loads whichever AudioClip you select in the Project window.</p> <p><b>3 Waveform</b> – click or drag to jump to that point and play from there.</p> <p><b>4 Playhead</b> – the current position, drawn over the waveform.</p> | <p><b>5 Play / Pause</b> – ► plays from the current position; ■■ pauses and keeps the position.</p> <p><b>6 Stop</b> – ■ stops and rewinds to the start.</p> <p><b>7 Time</b> – current position / total length, as minutes:seconds.</p> <p><b>8 Format</b> – sample rate of the loaded clip and “mono”.</p> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## How clips arrive in Preview

- **After Generate.** The new clip is loaded automatically and the Preview section is expanded.
- **From the Voice Library.** Clicking a voice loads its sample and starts playing.
- **From the Project window.** With **Follow** on, selecting any AudioClip loads it. Turn Follow off to keep a clip loaded while you browse.
- **By hand.** Drag a clip into the field, or use the picker.

Whenever the clip changes, everything tied to the old clip is reset: the trim range, the suggested output names, the playhead and the equalizer preview.

## What can be read

**.wav** files are decoded straight from disk, so the waveform and tools work no matter how the clip is imported. Any other format is read through Unity, which needs the clip’s import Load Type set to *Decompress On Load*; otherwise a warning explains how to fix it. Multi-channel audio is averaged to mono for the waveform and for every tool.

If the Editor’s own player cannot be reached, the transport buttons are disabled and a red “Editor playback unavailable” note gives the reason. The waveform and the audio tools keep working (section 12.9).

# 09

WINDOW REFERENCE

## Audio Tools

Trim, join and equalize audio without leaving Unity. Every tool saves a new asset and leaves the original alone.

Open **Audio Tools** to reach three tools. They share the same rules:

| Rule             | Detail                                                                                                                                             |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Input</b>     | The Trimmer and Equalizer work on the clip loaded in Preview. The Joiner has its own clip list.                                                    |
| <b>Output</b>    | A new <b>.wav</b> in the Output Folder, imported and highlighted in the Project window.                                                            |
| <b>Format</b>    | Mono at the clip's sample rate (the Joiner uses the highest rate in its list). Uses 16-bit PCM unless <b>32-bit float WAV</b> is ticked in Output. |
| <b>Originals</b> | Never modified. If the name you choose exists, a unique name is used instead.                                                                      |

### 9.1 Audio Trimmer

Cut a clip down to the part you want, for example to remove leading silence or a breath.

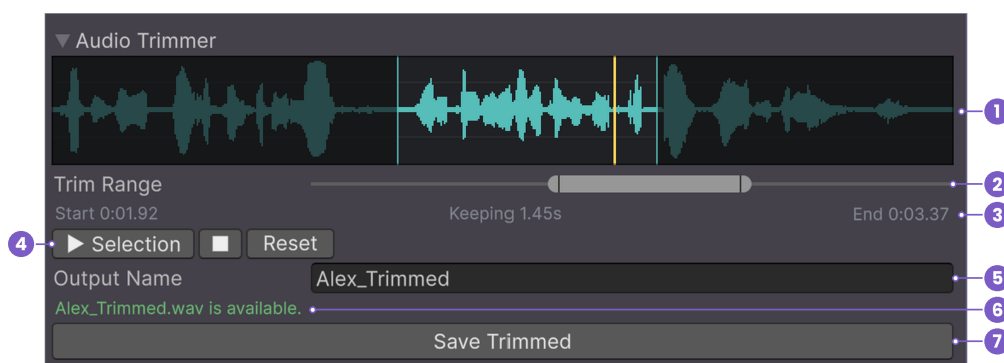


**Figure 7.** The Audio Trimmer keeping 1.45 seconds (from 0:01.92 to 0:03.37 in the readouts) of a clip. The dimmed parts of the waveform will be discarded.

- 1 Waveform** – the same waveform as Preview. Teal edges mark the kept region; dimmed areas are discarded. Click to scrub.
  - 2 Trim Range** – drag either handle to set the start and end.
  - 3 Readouts** – Start time, length being kept, End time.
  - 4 Transport** – ▶ **Selection** plays only the kept region; ■ stops; **Reset** selects the whole clip.
  - 5 Output Name** – defaults to `<clip>_Trimmed`.
  - 6 Availability** – green when the name is free; otherwise a unique name will be used.
  - 7 Save Trimmed** – writes the kept region as a new asset.
- ▶ **Selection** starts at the start handle and stops at the end handle. The stop point is read live, so you can move the end handle while it plays.
  - Clicking the waveform inside the trimmer plays to the end handle; clicking past it plays to the end of the clip.

- The kept region can never be empty: the handles keep a minimum gap of 0.1% of the clip.
- The status box confirms the saved path and length, and states that the original is unchanged.

## 9.2 Audio Joiner

Concatenate several clips into one, in list order, with an optional gap of silence between them. Useful for assembling dialogue or stitching together long text generated in pieces.

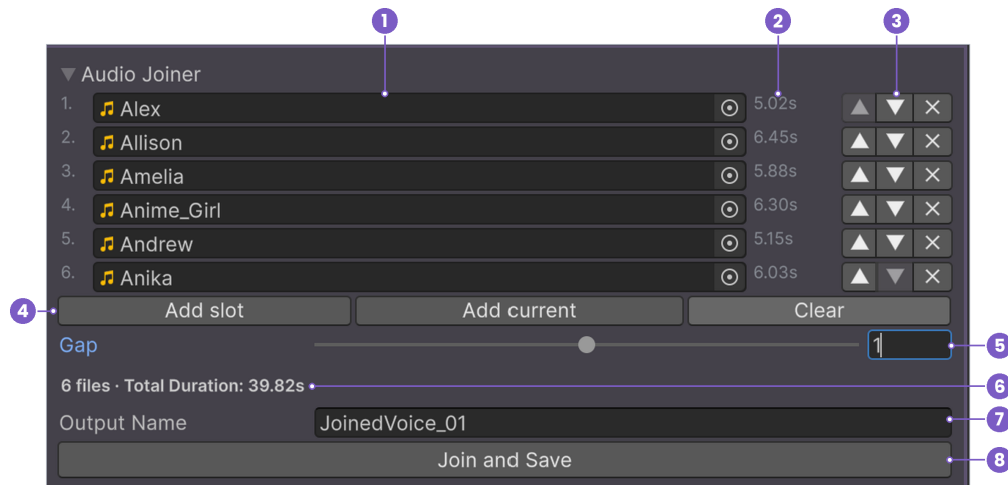
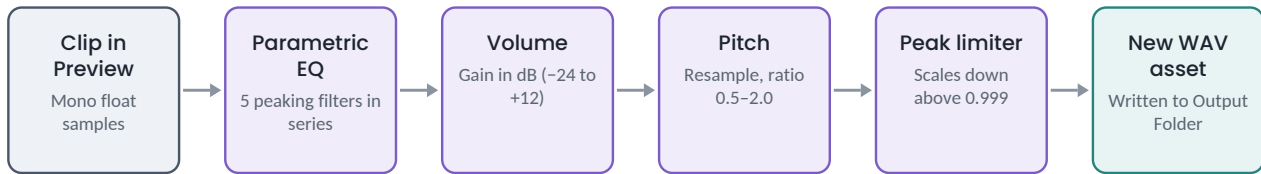


**Figure 8.** The Audio Joiner with six clips and a one-second gap.

- 1 Clip slot** – drop an AudioClip in each slot. Clips play top to bottom.
  - 2 Duration** – length of each clip, or “empty” for an unfilled slot.
  - 3 Order and remove** – ▲ moves earlier, ▼ moves later, × removes. The first ▲ and last ▼ are disabled.
  - 4 List buttons** – **Add slot** adds an empty row, **Add current** appends the Preview clip, **Clear** empties the list.
  - 5 Gap** – silence between consecutive clips, 0–2 seconds.
  - 6 Summary** – number of files and total duration, including gaps.
  - 7 Output Name** – defaults to `JoinedVoice_01`; the number advances after each save.
  - 8 Join and Save** – enabled with at least two clips and a name.
- Total duration = the clips’ lengths + gap × (number of clips – 1). In the screenshot, six clips of roughly 5–6.5 seconds plus five one-second gaps come to 39.82 s. The readout adds exact clip lengths, so it can differ from a sum of the rounded figures by a hundredth of a second.
  - Clips with different sample rates are resampled to the **highest** rate in the list with the same band-limited resampler used for voice cloning. The status box says when this happened.
  - Empty slots are skipped. Fewer than two readable clips shows “At least two readable clips are needed.”

## 9.3 Audio Equalizer

Adjust loudness, pitch and tone. The equalizer is real DSP: each band is a peaking filter, and the curve above the sliders is calculated from those same filters rather than drawn by hand.



*Preview and Apply and Save both run this exact chain, so what you hear is what gets saved.*

**Figure 9.** The processing chain, in order. The limiter only ever turns the level down, never up.

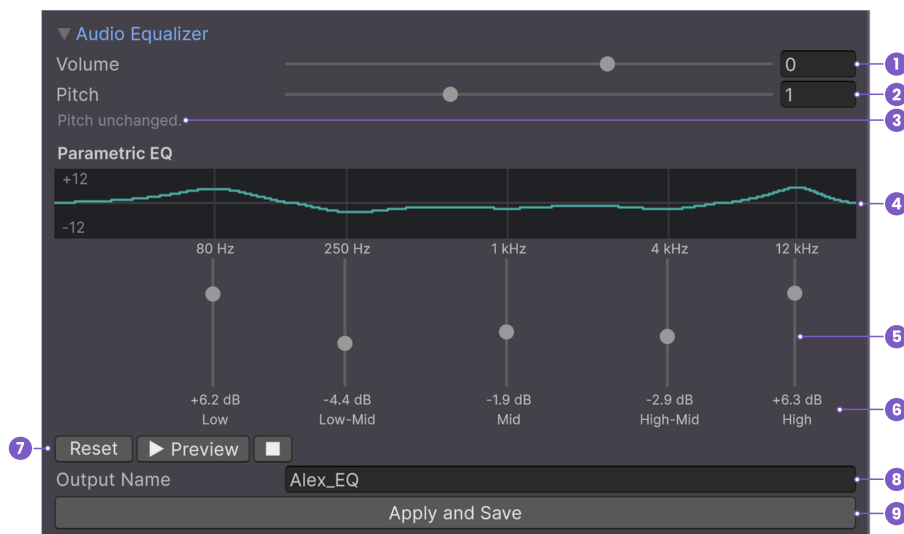


**Figure 10.** The Audio Equalizer with a “smile” curve: boosted lows and highs, gently scooped mids.

- 1 **Volume** – overall gain from -24 to +12 dB. 0 dB leaves the level unchanged.
- 2 **Pitch** – playback-rate ratio from 0.5 to 2.0.
- 3 **Pitch readout** – “Pitch unchanged.”, or the shift in semitones and the resulting length.
- 4 **Response curve** – the combined effect of all five bands, from 20 Hz upwards on a log axis.
- 5 **Band sliders** – five vertical sliders, each at its own frequency’s position on the curve.
- 6 **Band readouts** – gain in dB and the band name.
- 7 **Reset, Preview, Stop** – **Reset** makes everything neutral; **Preview** plays the processed result; **Stop** stops.
- 8 **Output Name** – defaults to `<clip>_EQ`.
- 9 **Apply and Save** – writes the processed clip as a new asset.

### The five bands

| Band     | Centre | Range  | Notes                   |
|----------|--------|--------|-------------------------|
| Low      | 80 Hz  | ±12 dB | Body and warmth.        |
| Low-Mid  | 250 Hz | ±12 dB | Boxiness and muddiness. |
| Mid      | 1 kHz  | ±12 dB | Presence.               |
| High-Mid | 4 kHz  | ±12 dB | Clarity and sibilance.  |
| High     | 12 kHz | ±12 dB | Air.                    |

Every band is a peaking filter with a fixed Q of 1.0. A slider within  $\pm 0.35$  dB of zero snaps to exactly 0, so a band can be reliably switched off, and bands at 0 dB are skipped entirely. A filter whose centre frequency would be at or above half the sample rate is ignored.

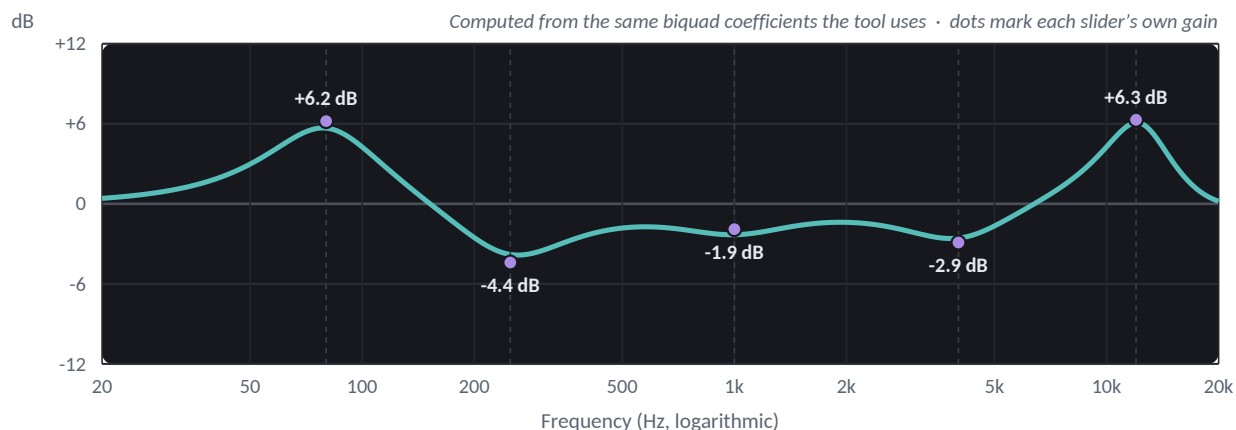


**Figure 11.** The response of the settings in the screenshot: +6.2, -4.4, -1.9, -2.9 and +6.3 dB. Individual bands overlap, so the curve does not pass exactly through each dot.

## Pitch is a tape-speed change

Pitch resamples the audio, exactly like changing tape speed: raising the pitch also **shortens** the clip, and lowering it lengthens it. It is not a duration-preserving pitch shifter. The readout under the slider shows both effects.

| Pitch ratio | Shift           | Length    |
|-------------|-----------------|-----------|
| 0.50        | -12.0 semitones | ×2.00     |
| 0.75        | -5.0 semitones  | ×1.33     |
| 0.90        | -1.8 semitones  | ×1.11     |
| 1.00        | none            | unchanged |
| 1.25        | +3.9 semitones  | ×0.80     |
| 1.50        | +7.0 semitones  | ×0.67     |
| 2.00        | +12.0 semitones | ×0.50     |

## Previewing and saving

► **Preview** and **Apply and Save** call the same processing function, so what you hear is exactly what is saved. The result is peak-limited so it cannot clip: if the processed peak went past full scale, the clip is scaled down to just under it and the status box says so.

### WATCH OUT

To make previewing reliable, ► **Preview** writes a scratch file called `_EqPreview.wav` into the Output Folder and overwrites it on every preview. It is safe to delete once you are done. Press **Apply and Save** to keep a named copy.

Practical sequences that combine the features in this guide.

## Record a single voice-over line

- 1 Choose a voice and type the line**  
Pick a voice in the library and type the text.
- 2 Generate until it sounds right**  
Leave **Random** on and generate a few takes. Each saves as the next numbered file.
- 3 Trim the edges**  
Select the best take in the Project window (with Follow on it loads into Preview), set the Trim Range and press **Save Trimmed**.
- 4 Use it**  
The trimmed WAV is a normal AudioClip. Drag it onto an AudioSource, a Timeline track or a Playable Director.

## Build a dialogue scene from several speakers

- 1 Generate each line separately**  
Switch voice and text between generations. Name files so they sort in story order, for example `Scene1_01`, `Scene1_02`.
- 2 Open the Audio Joiner**  
Press **Add slot** for each line and drag the clips in, or use **Add current** after generating each line.
- 3 Order and space them**  
Use ▲ ▼ to reorder and the Gap slider for the pause between speakers.
- 4 Join and Save**  
Name the result and press **Join and Save**. Different sample rates are handled automatically.

### TIP

Long passages are best generated as several sentences or paragraphs and then joined. Each clip stays well inside the 2048-position context window, and you can re-generate only the line you do not like.

## Get a take back later

- Note the seed printed in the status box when you like a take.
- Later, turn **Random** off, type that seed and keep the same text, voice and parameters.
- To explore alternatives, change only the seed and generate again.

## Make a new character voice

- 1 Prepare a reference**  
Record or find a clean clip of one speaker, 0.5–30 seconds. Trim it with the Audio Trimmer if needed.
- 2 Clone it**  
Add the clip and its exact transcript under **Clone a Voice**, name it and press **Clone Voice**.
- 3 Audition**  
Type a test line and generate. If the voice drifts, check the transcript against the audio.
- 4 Shape the tone**  
Use the Equalizer to warm or brighten the result. A pitch of 0.9, for instance, lowers it by about 1.8 semitones and makes the clip about 11% longer.

## Use the audio in your game

Generated clips are standard AudioClip assets, so anything that accepts audio works: AudioSource components, Timeline, UnityEvents, FMOD or Wwise imports. For speech generated while the game runs, use the scripting API described in Chapter 13.

# Settings, Persistence and Files

What the window remembers, what it writes to disk, and how to put everything back to defaults.

## What the window remembers

Two kinds of storage hold the window's state. **EditorPrefs** are shared by every project on the machine; the window's own serialized fields are saved by Unity with the editor layout.

### EditorPrefs keys

| Key                      | Remembers                   | Default (never stored)             |
|--------------------------|-----------------------------|------------------------------------|
| DeepVoice.ModelDirectory | Model Folder                | Assets/DeepVoice Offline/Models    |
| DeepVoice.OutputFolder   | Output Folder               | Assets/DeepVoice Offline/Generated |
| DeepVoice.VoicesFolder   | VoiceData Folder            | Assets/DeepVoice Offline/VoiceData |
| DeepVoice.FileName       | Output file name            | GeneratedVoice_01                  |
| DeepVoice.Threads        | CPU threads                 | 4                                  |
| DeepVoice.VoiceGuid      | Selected voice (asset GUID) | none                               |

A value is written only when it differs from its default, so a window left at defaults stores nothing and always follows whatever the current defaults are. The default model folder is deliberately never saved: an absolute path from one project would otherwise be restored in the next and override that project's own default. The selected voice is kept even when none is currently selected, so a voice that is briefly unavailable during an import is not forgotten.

### Saved with the editor layout

The text, all generation parameters, seed and Random, the 32-bit float option, which sections are expanded, the Preview clip and Follow toggle, the trim range, the Joiner list and gap, the equalizer settings and the Clone a Voice fields.

#### NOTE

Earlier builds used `Assets/DeepVoice_Local/Generated` and `Assets/DeepVoice_Local/Voices` as default folders. If a saved value still equals one of those, it is treated as “never changed” and replaced by the new default.

## Resetting everything

**Diagnostics Reset Saved Settings** clears every key above, restores every field to its first-launch value (text, parameters, folders, file name, threads, selected voice, section visibility, Joiner list, equalizer, Clone fields, Preview) and refreshes the library. It also removes the keys the scene panel stores in PlayerPrefs under `DeepVoice.Scene`. (section 4.6). Generated audio and voice profiles are never touched. It is unavailable while work is running.

## Files the tool writes

| Location                                               | File                   | Purpose                                                                                                                |
|--------------------------------------------------------|------------------------|------------------------------------------------------------------------------------------------------------------------|
| Library/DeepVoice_Local/                               | tokenizer_<hash>.bin   | Binary cache of the parsed tokenizer. Rebuilt automatically if tokenizer.json changes (checked by size and timestamp). |
| Library/DeepVoice_Local/                               | model_inspection.txt   | Report from <b>Inspect model files</b> .                                                                               |
| Library/DeepVoice_Local/ProfileB<br>ackup/<timestamp>/ | copies of .asset files | Backups made before <b>Repair voice profiles</b> changes anything.                                                     |
| Output Folder                                          | <name>.wav             | Generated, trimmed, joined and equalized clips.                                                                        |
| Output Folder                                          | _EqPreview.wav         | Scratch file for the equalizer preview. Safe to delete.                                                                |
| VoiceData Folder                                       | <voice>.asset          | Voice profile assets.                                                                                                  |

### HOW IT WORKS

The `Library` folder is Unity's local cache and is normally excluded from version control. Nothing in it is required: the tokenizer cache rebuilds on the next load.

## Model lifecycle

- **Loaded** on the first generation, the first voice preview that needs decoding, or when you press **Load Model**.
- **Shared**. The window, Voice Library and Inspector all use one copy through `DeepVoiceRuntimeHolder`, so the model is never loaded twice.
- **Kept** when you close the window.
- **Reloaded** if the Model Folder changes.
- **Released** before every assembly reload (a script recompile, or entering Play Mode with domain reload) and when the Editor quits, because the native sessions cannot outlive the managed wrapper. The next generation loads it again.
- **Released** on demand with **Unload Model**.

The architecture and inference pipeline behind the window, for developers who want to understand or extend it.

## 12.1 Architecture

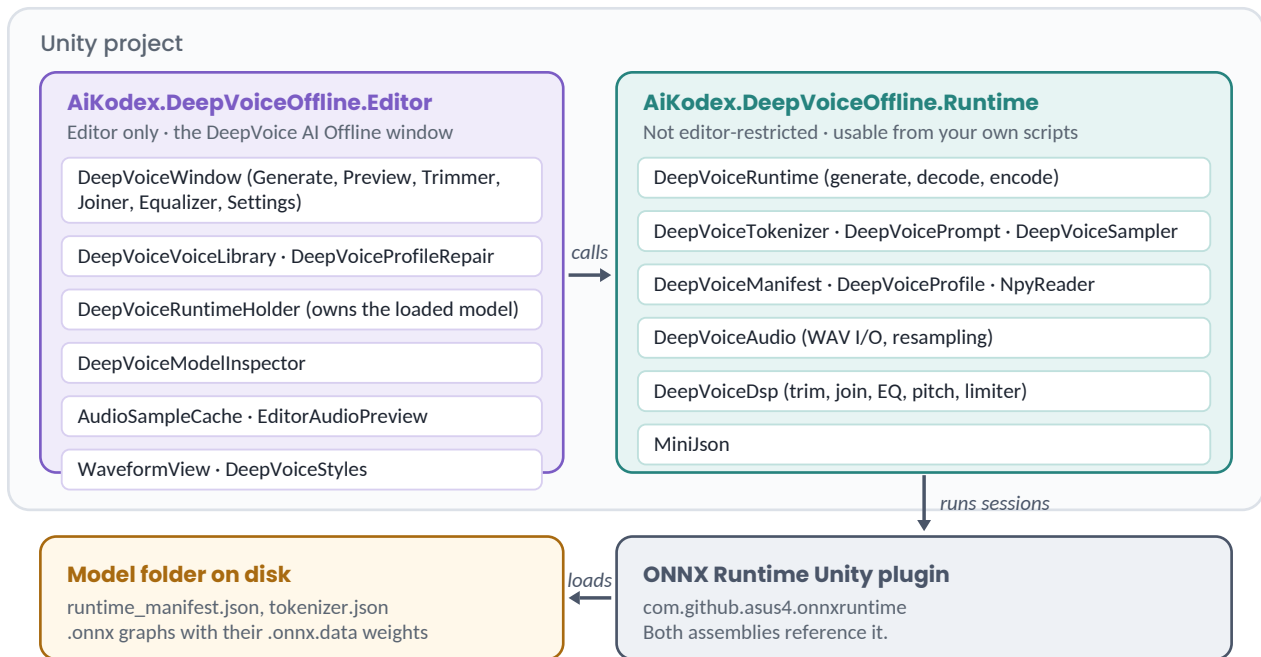


**Figure 12.** The two assemblies and what they depend on. The editor assembly depends on the runtime assembly, never the other way round.

The **runtime assembly** holds the inference engine, tokenizer, prompt builder, sampler, audio I/O and DSP. It is not restricted to the Editor, so the same code can be called from your own scripts. The **editor assembly** holds the window and everything that needs the Editor: the voice library, waveform drawing, playback and the diagnostic tools. Both reference the ONNX Runtime Unity plugin.

A single static class, `DeepVoiceRuntimeHolder`, owns the loaded model so that the window, the Voice Library and the Inspector share one copy. It frees the model before every assembly reload and when the Editor quits.

## 12.2 The generation pipeline

The runtime is a C# port of a Python reference implementation (`runtime.py` and `prompt.py`) onto ONNX Runtime. It uses three ONNX graphs for synthesis:

### 1 Clean and tokenize

The text is cleaned, then turned into token ids by a byte-level BPE tokenizer read from `tokenizer.json`.

- 2 Pack the prompt**  
 Instruction text, the reference transcript, the reference codes and your text are laid out in one  $[1, 11, \text{length}]$  tensor (next section).
- 3 Prefill the Slow AR**  
 The whole prompt is run through the Slow AR graph once, filling its key/value cache. The logits and hidden state of the last position are kept.
- 4 Sample a semantic token**  
 One token is drawn from the 4097 logits: 4096 semantic tokens plus a stop token. The stop token ends generation.
- 5 Run the Fast AR for the frame**  
 The semantic token becomes codebook 0. The Fast AR graph, primed from the Slow AR's hidden state, then samples codebooks 1 to 9 one after another.
- 6 Feed the frame back**  
 The frame (the semantic token plus its codes) becomes the next input column for the Slow AR, and steps 4–6 repeat until the stop token or the length cap.
- 7 Decode**  
 All frames go through the codec decoder in one call, producing the 44.1 kHz waveform.

## 12.3 Prompt layout

|            | Prefix<br>system header, reference transcript,<br>"Speech:" | Reference frames<br>one column per codec frame<br>of the reference | Suffix<br>user header, your text, assistant<br>header + voice marker | Generated<br>appended step by step as<br>audio frames |
|------------|-------------------------------------------------------------|--------------------------------------------------------------------|----------------------------------------------------------------------|-------------------------------------------------------|
| row 0      | text token ids                                              | code + semantic_begin_id                                           | text token ids                                                       | semantic token                                        |
| codebook 0 | 0                                                           | reference codes                                                    | 0                                                                    | 10 codes                                              |
| codebook 1 | 0                                                           | reference codes                                                    | 0                                                                    | 10 codes                                              |
| codebook 2 | 0                                                           | reference codes                                                    | 0                                                                    | 10 codes                                              |
|            |                                                             | ⋮                                                                  |                                                                      |                                                       |
| codebook 9 | 0                                                           | reference codes                                                    | 0                                                                    | 10 codes                                              |

**Shape:**  $[1, \text{numCodebooks} + 1, \text{length}]$ : 11 rows by (prefix + reference frames + suffix) columns.

The prompt must stay below `max_seq_len` (2048). Generated frames are appended after it, so a longer prompt leaves less room for audio.

**Figure 13.** The packed prompt. Row 0 carries token ids; rows 1–10 carry the ten codebooks, which are only filled where the reference recording sits.

## PROMPT TEMPLATE (ROW 0, IN READING ORDER)

```

<|im_start|>system
convert the provided text to speech reference to the following:

Text:
<|speaker:0|>{reference transcript}

Speech:
{reference frames}<|im_end|>
<|im_start|>user
{your text}<|im_end|>
<|im_start|>assistant
<|voice|>

```

A `<|speaker:0|>` tag is added in front of the reference transcript unless it already contains a `<|speaker:N|>` tag. Each reference frame appears in row 0 as its first-codebook code plus `semantic_begin_id`, and in rows 1–10 as its ten raw codes.

## 12.4 Model geometry

These values come from `runtime_manifest.json`. The figures shown are the defaults the loader uses when a key is absent, and describe the model the tool was built for.

| Part                     | Properties                                                                                                                                          |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Slow AR</b>           | 24 layers; 2 key/value heads of dimension 64; context of 2048 positions; outputs 4097 logits (4096 semantic + 1 stop) and an 896-wide hidden state. |
| <b>Fast AR</b>           | 4 layers; 2 key/value heads of dimension 64; works across the 10 codebooks of a frame; outputs 4096-way logits.                                     |
| <b>Codec</b>             | 10 codebooks of 4096 entries; 2048-sample hop at 44.1 kHz (about 21.5 frames per second).                                                           |
| <b>Vocabulary</b>        | 155,776 tokens. Semantic tokens occupy ids 151678–155773; the stop token is <code>&lt; im_end &gt;</code> (151645).                                 |
| <b>Default precision</b> | int4 for the Slow and Fast AR graphs, fp16 for the codec.                                                                                           |

## 12.5 Key/value caches

Autoregressive transformers reuse the attention keys and values of earlier positions. The runtime keeps these in persistent arrays pinned behind ONNX tensors for the life of the session, so each step only has to scatter the newly computed slice into place; nothing is reallocated per token. Slow caches have shape [1, heads, 2048, 64] per key and per value, one pair for each of the 24 layers; fast caches are [1, heads, 10, 64] for each of the 4 layers. The element type (fp16 or fp32) is read from the graph. The Slow cache is cleared at the start of every generation and the Fast cache at the start of every frame.

Output shapes are checked as they are read. If a graph returns something the manifest does not describe, the error names the tensor and its shape and tells you to run **Inspect model files**, instead of failing with an out-of-range copy.

## 12.6 Tokenizer

`DeepVoiceTokenizer` reproduces the HuggingFace `encode(text, add_special_tokens=False)` behaviour: special (added) tokens are split out first by longest match using a trie; the remaining text is pre-tokenized with the regular expression from `tokenizer.json` (a Qwen-style default is used if none is present); each piece is mapped through the GPT-2 byte-to-unicode table; BPE merges are applied in rank order; and the resulting symbols are looked up in the vocabulary. Both the legacy `"left right"` and the newer `["left", "right"]` merge formats are accepted. Because the 12 MB JSON is slow to parse, the parsed tables are stored in a compact binary cache that is checked against the source file's size and timestamp.

## 12.7 Audio I/O and DSP

- **WAV reading.** PCM at 8, 16, 24 and 32 bits, IEEE float at 32 and 64 bits, and `WAVE_FORMAT_EXTENSIBLE`. Channels are averaged to mono.
- **WAV writing.** Mono 16-bit PCM (clamped to  $\pm 1$ , NaN written as silence) or mono 32-bit float.
- **Resampling.** A band-limited windowed-sinc converter (Blackman window, 24 taps either side). The cutoff drops below Nyquist when downsampling so the result does not alias.
- **Equalizer.** Direct-form-1 biquads using the Audio EQ Cookbook peaking formulas, run in series.
- **Limiter.** A simple peak normaliser that only reduces level, to a 0.999 ceiling.

PEAKING FILTER USED BY EVERY EQUALIZER BAND

```
A = 10^(gain_dB / 40)          w0 = 2*pi*f / fs          alpha = sin(w0) / (2*Q)

b0 = 1 + alpha*A      b1 = -2*cos(w0)      b2 = 1 - alpha*A
a0 = 1 + alpha/A      a1 = -2*cos(w0)      a2 = 1 - alpha/A

y[n] = (b0*x[n] + b1*x[n-1] + b2*x[n-2] - a1*y[n-1] - a2*y[n-2]) / a0
```

## 12.8 Threading

Anything slow runs through one scheduler in the window. The main thread snapshots all the Unity data it needs (profile codes, folders, options), a background task does the work and reports progress through volatile fields, and a completion callback is run back on the main thread, where it may create assets and touch Unity objects. Unity assembly reloads are locked for the duration, and closing the window cancels the task and waits up to ten seconds for it to finish so the lock is always released.

## 12.9 Editor playback

Unity has no public API for playing audio in the Editor, so the tool reaches the Editor's own preview player (`UnityEditor.AudioUtil`) by reflection and tries the method names used before and after Unity 2020.2. The playhead is read back from the same player. If none of the expected methods exist, `EditorAudioPreview.Available` is false and `FailureReason` says why; the window then disables its transport buttons and shows an "Editor playback unavailable" note, while everything else (waveform, trimming, joining, equalizing, generation) keeps working. Trimmed or processed clips that have been saved play normally through Unity's regular import.

Call generation, voice encoding and the audio DSP from your own code. Everything lives in the `AiKodex.DeepVoiceOffline` namespace.

The runtime assembly `AiKodex.DeepVoiceOffline.Runtime` is not editor-restricted, so these types work in editor scripts and in code that runs while your game does. Add a reference to the assembly definition from your own.

## WATCH OUT

**Rules for safe use.** (1) Unity objects must not be created or read off the main thread: take a snapshot with `DeepVoiceProfile.ToData()` first and pass that to `Generate`. (2) `AudioClip.Create` and asset APIs run on the main thread. (3) A `DeepVoiceRuntime` reuses internal buffers, so run one `Generate` at a time per instance. (4) The model folder must be a real folder path that the process can read; in a player build that means copying the files somewhere such as `Application.persistentDataPath`. (5) Whether a given build target works also depends on the ONNX Runtime plugin supporting that platform.

## Generating speech

GENERATE A LINE OF SPEECH FROM A SCRIPT (ILLUSTRATIVE)

```
using System.Threading.Tasks;
using AiKodex.DeepVoiceOffline;
using UnityEngine;

public class SpeakExample : MonoBehaviour
{
    public DeepVoiceProfile voice;      // a profile from your VoiceData folder
    public string modelFolder;          // folder containing runtime_manifest.json
    public AudioSource output;

    DeepVoiceRuntime _runtime;          // load once, reuse for every line

    public async void Speak(string text)
    {
        DeepVoiceProfileData snapshot = voice.ToData();    // copy on the main thread
        string cache = Application.persistentDataPath;    // tokenizer cache folder

        var options = new DeepVoiceGenerationOptions
        {
            Text = text, MaxNewTokens = 512,
            Temperature = 0.7f, TopP = 0.9f, TopK = 50, Seed = 42,
        };

        DeepVoiceGenerationResult result = await Task.Run(() =>
        {
            if (_runtime == null)
                _runtime = new DeepVoiceRuntime(modelFolder, threads: 4,
                                                tokenizerCacheDirectory: cache);
            return _runtime.Generate(snapshot, options);
        });
    }
}
```

```

    // AudioClip.Create is a main-thread API, so it runs after the await.
    var clip = AudioClip.Create("tts", result.Samples.Length, 1,
                                result.SampleRate, false);
    clip.SetData(result.Samples, 0);
    output.PlayOneShot(clip);
}

void OnDestroy() => _runtime?.Dispose();
}

```

`Generate` also takes an optional `Action<int,int>` called after every frame with (frames so far, frame target) and a `CancellationToken`. Cancelling throws `OperationCanceledException`.

## Cloning a voice from a script

REGISTER A VOICE WITHOUT THE WINDOW (ILLUSTRATIVE, EDITOR ONLY)

```

using System.IO;
using AiKodex.DeepVoiceOffline;
using UnityEditor;
using UnityEngine;

public static class CloneFromScript
{
    public static DeepVoiceProfile Clone(string modelFolder, string wavPath,
                                         string transcript, string name)
    {
        string manifestPath = Path.Combine(modelFolder, "runtime_manifest.json");
        var manifest = DeepVoiceManifest.Load(manifestPath);

        float[] mono = DeepVoiceAudio.ReadWavMono(wavPath, out int rate);
        mono = DeepVoiceAudio.Resample(mono, rate, manifest.SampleRate);
        mono = DeepVoiceAudio.PadToMultiple(mono, manifest.CodecFrameSize);

        ushort[] codes = DeepVoiceRuntime.EncodeReferenceAudio(
            modelFolder, mono, manifest.NumCodebooks, threads: 4, out int frames);

        var profile = ScriptableObject.CreateInstance<DeepVoiceProfile>();
        profile.voiceName = name;
        profile.referenceText = transcript;
        profile.sourceSampleRate = rate;
        profile.modelFingerprint = manifest.ModelFingerprint;
        profile.SetCodes(codes, manifest.NumCodebooks, frames);

        string assetPath = $"Assets/DeepVoice Offline/VoiceData/{name}.asset";
        AssetDatabase.CreateAsset(profile, assetPath);
        AssetDatabase.SaveAssets();
        return profile;
    }
}

```

## Processing audio

THE EQUALIZER CHAIN AS CODE (ILLUSTRATIVE)

```
float[] samples = DeepVoiceAudio.ReadWavMono(path, out int rate);

DeepVoiceEqBand[] bands = DeepVoiceDsp.DefaultBands(); // 80, 250, 1k, 4k, 12k Hz
bands[0].GainDb = 4f;
bands[4].GainDb = 3f;

float[] result = DeepVoiceDsp.Process(samples, rate, bands,
                                     volumeDb: -2f, pitchRatio: 1f);
result = DeepVoiceDsp.LimitPeak(result);
DeepVoiceAudio.WriteWav16(outputPath, result, rate);
```

### NOTE

The code samples were written from the public signatures in the shipped scripts and are illustrative: they have not been compiled against a Unity project.

## Runtime types

### DeepVoiceRuntime (IDisposable)

| Member                                                                                                                                                                            | Description                                                                                                                                                                                        |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DeepVoiceRuntime(string modelDirectory, int threads = 0, string tokenizerCacheDirectory = null)                                                                                   | Loads the manifest, tokenizer and the three synthesis graphs. <code>threads</code> of 0 uses ONNX Runtime's default. A null cache directory disables the tokenizer cache.                          |
| DeepVoiceManifest Manifest { get; }                                                                                                                                               | The parsed runtime_manifest.json.                                                                                                                                                                  |
| DeepVoiceTokenizer Tokenizer { get; }                                                                                                                                             | The loaded tokenizer.                                                                                                                                                                              |
| string ModelDirectory { get; }                                                                                                                                                    | The folder the model was loaded from.                                                                                                                                                              |
| DeepVoiceGenerationResult Generate(DeepVoiceProfileData voice, DeepVoiceGenerationOptions options, Action<int,int> onFrame = null, CancellationToken cancellationToken = default) | Synthesizes speech. Throws <code>InvalidOperationException</code> for an unusable voice, a codebook mismatch, an over-long prompt or no output, and <code>ArgumentException</code> for empty text. |
| float[] DecodeVoiceCodes(DeepVoiceProfileData voice)                                                                                                                              | Decodes a profile's reference codes back to audio. This is what the Voice Library plays for profiles without a clip.                                                                               |
| static ushort[] EncodeReferenceAudio(string modelDirectory, float[] monoSamples, int numCodebooks, int threads, out int frames)                                                   | Runs the codec encoder over a mono 44.1 kHz recording (already padded to a multiple of the codec frame size). Creates and releases its own session.                                                |
| void Dispose()                                                                                                                                                                    | Releases all native sessions and buffers.                                                                                                                                                          |

## Generation data

| Member                                                                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DeepVoiceGenerationOptions                                                   | Fields: <code>Text</code> (""), <code>MaxNewTokens</code> (512), <code>Temperature</code> (0.7), <code>TopP</code> (0.9), <code>TopK</code> (50), <code>Seed</code> (42).                                                                                                                                                                                                                                                                                                 |
| DeepVoiceGenerationResult                                                    | <code>float[] Samples</code> , <code>int SampleRate</code> , <code>int Frames</code> , <code>int PromptTokens</code> , <code>bool HitTokenLimit</code> .                                                                                                                                                                                                                                                                                                                  |
| DeepVoiceProfile : ScriptableObject                                          | Fields <code>voiceName</code> , <code>referenceText</code> , <code>numCodebooks</code> , <code>frames</code> , <code>codes</code> , <code>sourceSampleRate</code> , <code>modelFingerprint</code> , <code>referenceClip</code> , <code>language</code> . Members: <code>IsValid</code> , <code>DescribeProblem()</code> , <code>ToData()</code> , <code>SetCodes(ushort[], int, int)</code> , <code>ToUShortCodes()</code> , <code>ApproximateSeconds(hop, rate)</code> . |
| DeepVoiceProfileData                                                         | Thread-safe plain copy: <code>Name</code> , <code>ReferenceText</code> , <code>NumCodebooks</code> , <code>Frames</code> , <code>Codes</code> , <code>DescribeProblem()</code> .                                                                                                                                                                                                                                                                                          |
| <code>NpyReader.ReadCodes(string path, out int rows, out int columns)</code> | Reads a 2D (or [1, rows, columns]) .npz code matrix. Supports <code>uint16</code> , <code>int16</code> , <code>int32</code> and <code>int64</code> in C order.                                                                                                                                                                                                                                                                                                            |

## Model description, text and sampling

| Member                                                                                                   | Description                                                                                                                                                                                                         |
|----------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>DeepVoiceManifest.Load(string path)</code>                                                         | Parses <code>runtime_manifest.json</code> . Properties: <code>SlowModelFile</code> , <code>FastModelFile</code> , <code>CodecDecoderFile</code> , <code>SecondsPerFrame</code> . All keys are listed in Appendix A. |
| <code>DeepVoiceTokenizer.Load(string path, string cacheDirectory = null)</code>                          | Loads <code>tokenizer.json</code> , using a binary cache when a directory is given.                                                                                                                                 |
| <code>List&lt;int&gt; Encode(string text) / void Encode(string text, List&lt;int&gt; destination)</code> | Token ids for a string; special tokens are matched literally.                                                                                                                                                       |
| <code>VocabCount</code> , <code>AddedTokenCount</code>                                                   | Sizes of the vocabulary and the special-token set.                                                                                                                                                                  |
| <code>DeepVoicePrompt.CleanText / FormatReferenceText / Build</code>                                     | Text cleaning, speaker-tag handling and construction of the packed prompt tensor.                                                                                                                                   |
| <code>DeepVoiceSampler.Sample / SampleSemantic; DeepVoiceRandom(long seed)</code>                        | Token sampling and its xoshiro256** random source ( <code>NextULong</code> , <code>NextDouble</code> ).                                                                                                             |
| MiniJson                                                                                                 | A small JSON parser: <code>Parse</code> , <code>Get</code> , <code>GetString</code> , <code>GetInt</code> , <code>GetDouble</code> , <code>GetBool</code> , <code>AsObject</code> , <code>AsArray</code> .          |

## Audio and DSP

| Member                                                                             | Description               |
|------------------------------------------------------------------------------------|---------------------------|
| <code>DeepVoiceAudio.WriteWav16(path, float[] samples, int sampleRate)</code>      | Writes mono 16-bit PCM.   |
| <code>DeepVoiceAudio.WriteWav32Float(path, float[] samples, int sampleRate)</code> | Writes mono 32-bit float. |

| Member                                                                                                                   | Description                                                                                                                                                   |
|--------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>DeepVoiceAudio.ReadWavMono(path, out int sampleRate)</code>                                                        | Reads a WAV and returns mono floats.                                                                                                                          |
| <code>DeepVoiceAudio.Resample(float[] input, int inputRate, int outputRate, int halfTaps = 24)</code>                    | Windowed-sinc sample-rate conversion.                                                                                                                         |
| <code>DeepVoiceAudio.PadToMultiple(float[] input, int multiple)</code>                                                   | Zero-pads to a multiple of a length.                                                                                                                          |
| <code>DeepVoiceDsp.Trim(samples, startSample, endSample)</code>                                                          | Copies the half-open range [start, end).                                                                                                                      |
| <code>DeepVoiceDsp.Join(IList&lt;float[]&gt; clips, IList&lt;int&gt; rates, int targetRate, float gapSeconds = 0)</code> | Resamples and concatenates clips with optional silence between them.                                                                                          |
| <code>DeepVoiceDsp.ApplyGain / ChangePitch / ApplyEq / Process</code>                                                    | Individual steps and the combined EQ → gain → pitch chain. <code>ChangePitch</code> clamps the ratio to 0.25–4.                                               |
| <code>DeepVoiceDsp.PeakAmplitude / WouldClip / LimitPeak / ComputePeaks</code>                                           | Level analysis, limiting to a ceiling (default 0.999) and min/max buckets for drawing waveforms.                                                              |
| <code>DeepVoiceDsp.DbToLinear / LinearToDb / RatioToSemitones / DefaultBands</code>                                      | Conversions and the default five-band layout.                                                                                                                 |
| <code>DeepVoiceEqBand, DeepVoiceBiquad</code>                                                                            | One EQ band ( <code>Frequency</code> , <code>GainDb</code> , <code>Q</code> ) and the peaking filter with <code>Process</code> and <code>MagnitudeDb</code> . |

## Editor-only helpers

These live in `AiKodex.DeepVoiceOffline.Editor` and are available in editor scripts only.

| Member                                                           | Description                                                                                                                                                                             |
|------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>DeepVoiceWindow.Open()</code>                              | Opens the window (the <b>Window</b> menu item). <code>DeepVoiceWindow.MaxTextLength</code> is 2000.                                                                                     |
| <code>DeepVoiceRuntimeHolder</code>                              | <code>GetOrLoad(directory, threads)</code> , <code>Unload()</code> , <code>IsLoaded</code> , <code>LoadedDirectory</code> , <code>CacheDirectory</code> . The shared model owner.       |
| <code>DeepVoiceVoiceLibrary</code>                               | <code>Entries</code> , <code>Refresh(folder)</code> , <code>MarkStale()</code> , <code>Find(profile)</code> , <code>ResolvedScope</code> , <code>Diagnosis</code> .                     |
| <code>DeepVoiceModelInspector.Inspect(directory, threads)</code> | Returns the text report that <b>Inspect model files</b> saves.                                                                                                                          |
| <code>DeepVoiceProfileRepair</code>                              | <code>Inspect(folder)</code> , <code>Repair(folder, out backup, out error)</code> , <code>CountAssetFiles(folder)</code> .                                                              |
| <code>AudioSampleCache</code>                                    | <code>Get(clip, out error)</code> , <code>Invalidate(clip)</code> , <code>Clear()</code> , <code>CreateClip(name, mono, rate)</code> .                                                  |
| <code>EditorAudioPreview</code>                                  | <code>Available</code> , <code>FailureReason</code> , <code>Play</code> , <code>Stop</code> , <code>IsPlaying</code> , <code>PositionSeconds</code> , <code>NormalizedPosition</code> . |

# 14

REFERENCE

## Troubleshooting

Find the message you are seeing, understand the cause and fix it. Wording matches what the tool displays.

### Setup and model

| Symptom or message                                                                                                    | Cause                                                       | Fix                                                                                                                                                    |
|-----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| runtime_manifest.json not found in this folder.                                                                       | Model Folder is wrong or empty.                             | Point it at the folder that contains <code>runtime_manifest.json</code> .                                                                              |
| Model file not found: '...'                                                                                           | A graph named by the manifest is missing.                   | Copy the missing <code>.onnx</code> (and its <code>.onnx.data</code> ) next to the manifest.                                                           |
| tokenizer.json not found at '...'                                                                                     | The <code>tokenizer/</code> folder is missing.              | Add <code>tokenizer/tokenizer.json</code> to the model folder.                                                                                         |
| Model error (red indicator)                                                                                           | Loading failed. The reason is in Diagnostics.               | Fix the cause, then press <b>Load Model</b> .                                                                                                          |
| ... does not divide into rows of N / KV<br>cache delta ... has shape ... / slow AR<br>graph has N outputs, expected M | The graphs do not match the manifest.                       | Run <b>Inspect model files</b> and compare the report with <code>runtime_manifest.json</code> . Use the model files and manifest that belong together. |
| Inspect report shows MISMATCH lines                                                                                   | A graph input or output differs from what the port expects. | The report names each mismatch. The port reads graph names literally, so graphs must be exported with the documented names.                            |
| A changed thread count has no effect                                                                                  | The model is already loaded.                                | Press <b>Unload Model</b> , then generate or press <b>Load Model</b> .                                                                                 |
| Editor will not recompile while I generate                                                                            | Assembly reloads are locked during work.                    | Wait for it to finish, or press <b>Cancel</b> .                                                                                                        |

### Voices

| Symptom or message                                                    | Cause                                    | Fix                                                                               |
|-----------------------------------------------------------------------|------------------------------------------|-----------------------------------------------------------------------------------|
| No voice profiles found. This model is zero-shot...                   | The VoiceData folder has no profiles.    | Clone a voice (Chapter 7) or point VoiceData Folder at your profiles.             |
| The VoiceData folder '...' does not exist.                            | Folder path is wrong.                    | Create it, point the field at the right one, or clone a voice (which creates it). |
| N of the M .asset file(s) ... point at a script that no longer exists | Profiles reference an older script GUID. | Press <b>Repair voice profiles</b> . Originals are backed up first.               |
| ... already use the current script but did not load                   | Unity has not imported them.             | Right-click the folder and choose <i>Reimport</i> .                               |

| Symptom or message                                                                                                   | Cause                                         | Fix                                                                |
|----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|--------------------------------------------------------------------|
| ... none are DeepVoice voice profiles that Unity can read as text                                                    | Assets are binary-serialized or another type. | Switch Asset Serialization to Force Text and re-save, or re-clone. |
| Unusable: profile contains no codec frames / codec code array size does not match... / reference transcript is empty | The profile is incomplete or damaged.         | Re-clone the voice, or fill in the transcript in the Inspector.    |
| Voice 'x' has N codebooks but the model expects M.                                                                   | Voice and model do not match.                 | Use a voice registered with this model.                            |
| Reference audio must be 0.5-30 seconds, got X s.                                                                     | Clip length out of range.                     | Trim or choose another clip.                                       |
| codec_encoder_fp16.onnx not found...                                                                                 | Registration files missing.                   | Add the <code>registration/</code> folder to the model folder.     |
| 'x' has no reference recording, so its sample must be decoded by the codec. Set the model folder first.              | Voice has no clip and no model folder is set. | Set Model Folder.                                                  |

## Generating

| Symptom or message                                                  | Cause                                                   | Fix                                                                                                          |
|---------------------------------------------------------------------|---------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| Generate is greyed out                                              | A requirement is missing.                               | Read the red message beneath it; see section 5.7.                                                            |
| Prompt is N positions, which exceeds the model limit of 2048...     | Reference plus text is too long for the context window. | Use a shorter reference recording or split the text.                                                         |
| The model produced no codec frames...                               | The model stopped immediately.                          | Try another seed, longer text or a different voice, and check the profile's transcript.                      |
| Stopped at the frame cap                                            | Generation reached Maximum Length.                      | Raise <b>Maximum Length</b> (the message calls it Max Frames), or split the text.                            |
| Speech sounds wrong, garbled or off-voice                           | Poor reference, wrong transcript or an unlucky take.    | Check the transcript word for word, use a cleaner reference, then try new seeds or a lower Output Variation. |
| First generation is slow                                            | The model (about 1 GB) and tokenizer are loading.       | Later generations reuse them. Keep the model loaded, and raise CPU threads up to your physical core count.   |
| Wrote ... but Unity did not import it as an AudioClip.              | Import failed.                                          | Check the Console, then reimport the file.                                                                   |
| Output folder must be inside Assets so Unity can import the result. | Folder is outside the project.                          | Choose an Output Folder under <b>Assets</b> .                                                                |
| ... already exists and would be overwritten.                        | Name is taken.                                          | Press <b>+1</b> , or accept the overwrite.                                                                   |
| Audio crackles or distorts                                          | Output exceeded full scale.                             | Tick <b>32-bit float WAV</b> , or lower the level with the Equalizer's Volume.                               |
| Cancelled.                                                          | You pressed Cancel.                                     | Nothing was saved; generate again.                                                                           |

## Playback and audio tools

| Symptom or message                                                                                | Cause                                                 | Fix                                                                    |
|---------------------------------------------------------------------------------------------------|-------------------------------------------------------|------------------------------------------------------------------------|
| Editor playback unavailable: ...                                                                  | The Editor's internal preview API could not be found. | The waveform and tools still work; saved clips play normally in Unity. |
| Unity would not decode 'x'. Set its import Load Type to 'Decompress On Load', or use a .wav file. | A compressed clip set to stream from disk.            | Change the import setting or use a WAV.                                |
| Could not read 'x.wav': ...                                                                       | Unsupported or damaged WAV.                           | Re-export as PCM or float WAV.                                         |
| The trim range is empty.                                                                          | Start and end are the same.                           | Move the handles apart.                                                |
| At least two readable clips are needed.                                                           | Joiner has fewer than two valid clips.                | Add more clips.                                                        |
| Processing produced no audio.                                                                     | The Equalizer preview returned nothing.               | Reload the clip and try again.                                         |
| Pitch changes the clip length                                                                     | Pitch is a tape-speed change by design.               | See section 9.3.                                                       |
| Leftover _EqPreview.wav                                                                           | Equalizer preview scratch file.                       | Delete it.                                                             |

The boundaries to design around, and quick answers to common questions.

## Limits at a glance

| Item                       | Limit                                                                    |
|----------------------------|--------------------------------------------------------------------------|
| Text length                | 2000 characters                                                          |
| Languages                  | EN, ES, FR, DE, IT, NL, PL, JA, KO, ZH, YUE                              |
| Reference recording        | 0.5–30 seconds, one speaker                                              |
| Context window             | 2048 positions shared by the prompt and the generated frames             |
| Maximum Length slider      | 32–2000 frames (about 1.5–92.9 s), further limited by the context window |
| Output                     | Mono, 44.1 kHz; 16-bit PCM or 32-bit float WAV                           |
| CPU threads                | 1–16                                                                     |
| Volume / EQ band / Pitch   | –24 to +12 dB / $\pm 12$ dB per band / ratio 0.5–2.0                     |
| Joiner gap / minimum clips | 0–2 seconds / 2 clips                                                    |
| Voice grid                 | 1–4 columns depending on window width                                    |

## Frequently asked questions

### Does it need an internet connection?

No. The scripts contain no network calls. All processing happens locally through ONNX Runtime.

### Does it use the GPU?

No. The sessions are created with default options and no GPU execution provider, so inference runs on the CPU.

### Can I use it in a built game?

The runtime assembly is not restricted to the Editor, so the engine can be called from your own scripts (Chapter 13). The window itself is editor-only. A player build also needs a platform the ONNX Runtime plugin supports and the model files in a readable folder.

### How do I get exactly the same audio again?

Keep the text, voice and parameters, turn Random off and enter the seed from the status message (Chapter 6).

### Why did my clip end early or get cut off?

Either the model finished the sentence, or it reached Maximum Length. When the cap is hit the status message says so. Long references also reduce the room available.

**Why is the first generation slower?**

The model and tokenizer load on first use. After that the model stays in memory until an assembly reload, Unload Model or quitting the Editor.

**Why does changing pitch change the length?**

Pitch is a resample, like changing tape speed. It is not a duration-preserving pitch shifter.

**Is the output stereo?**

No. Output is mono, and the tools mix multi-channel input down to mono.

**Do the audio tools change my original clips?**

Never. Each tool saves a new asset. The one scratch file is `_EqPreview.wav`.

**Where did my files go?**

Into the Output Folder (default `Assets/DeepVoice Offline/Generated`). The clip is also highlighted in the Project window right after saving.

**Can I move voice profiles between projects?**

Yes. Copy the `.asset` (and optionally its clip). They only work with a model that has the same number of codebooks.

**Is there a UI I can use in a scene?**

The package shows a compact scene panel (Chapter 4) for basic generation. For a UI of your own, or to speak while the game runs, call the runtime through the scripting API (Chapter 13).

## A

APPENDIX

## runtime\_manifest.json

Every key the loader reads, with the default used when a key is missing.

| Key                     | Default             | Use                                                                                        |
|-------------------------|---------------------|--------------------------------------------------------------------------------------------|
| model_family            | "deepvoice"         | Read; not used by the current runtime.                                                     |
| activation_dtype        | "float16"           | Read; not used (tensor types are read from the graphs).                                    |
| slow_logits_layout      | "semantic_then_eos" | Read; not used.                                                                            |
| slow_logits_size        | 4097                | Width of the Slow AR logits (4096 semantic + 1 stop).                                      |
| max_seq_len             | 2048                | Context window; sizes the Slow KV cache.                                                   |
| num_layers              | 24                  | Slow AR layers.                                                                            |
| num_fast_layers         | 4                   | Fast AR layers.                                                                            |
| num_codebooks           | 10                  | Codebooks per frame.                                                                       |
| n_local_heads           | 2                   | Slow AR key/value heads.                                                                   |
| fast_n_local_heads      | 2                   | Fast AR key/value heads.                                                                   |
| head_dim                | 64                  | Slow AR head dimension.                                                                    |
| fast_head_dim           | 64                  | Fast AR head dimension.                                                                    |
| fast_dim                | 896                 | Width of the Slow AR hidden state passed to the Fast AR.                                   |
| vocab_size              | 155776              | Read; not used.                                                                            |
| codebook_size           | 4096                | Entries per codebook; sizes the Fast AR logits.                                            |
| semantic_begin_id       | 151678              | First semantic token id.                                                                   |
| semantic_end_id         | 155773              | Last semantic token id.                                                                    |
| eos_token_id            | 151645              | Read; not used (the stop token is im_end_id).                                              |
| pad_token_id            | 151643              | Read; not used.                                                                            |
| im_end_id               | 151645              | Stop token.                                                                                |
| sample_rate             | 44100               | Output rate and the rate references are resampled to.                                      |
| codec_frame_size        | 2048                | References are padded to a multiple of this before encoding.                               |
| codec_hop_length        | 2048                | Used for <code>SecondsPerFrame</code> .                                                    |
| default_precision       | "int4"              | Selects <code>slow_ar*.onnx</code> and <code>fast_ar*.onnx</code> .                        |
| default_codec_precision | "fp16"              | Selects the entry of <code>codec_models</code> .                                           |
| codec_models            | { "fp16": ... }     | Maps a codec precision to its file. The fallback is { "fp16": "codec_decoder_fp16.onnx" }. |
| model_fingerprint       | " "                 | Copied into new voice profiles.                                                            |

# B

APPENDIX

## Defaults at a Glance

The values the window starts with, and what Reset Saved Settings restores.

| Setting                                              | Default                                                                                                                                                               |
|------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Text                                                 | "Welcome to DeepVoice AI Offline, running on device inside Unity."                                                                                                    |
| Maximum Length                                       | 512 frames                                                                                                                                                            |
| Output Variation / Creativity Range / Choice Variety | 0.7 / 0.9 / 50                                                                                                                                                        |
| Seed / Random                                        | 42 / on                                                                                                                                                               |
| CPU threads                                          | 4                                                                                                                                                                     |
| Model Folder                                         | Assets/DeepVoice Offline/Models                                                                                                                                       |
| Output Folder                                        | Assets/DeepVoice Offline/Generated                                                                                                                                    |
| VoiceData Folder                                     | Assets/DeepVoice Offline/VoiceData                                                                                                                                    |
| File Name                                            | GeneratedVoice_01                                                                                                                                                     |
| 32-bit float WAV                                     | off                                                                                                                                                                   |
| Clone a Voice: name / language                       | speaker_a / EN                                                                                                                                                        |
| Joiner: output name / gap                            | JoinedVoice_01 / 0 s                                                                                                                                                  |
| Equalizer: volume / pitch / bands                    | 0 dB / 1.0 / all 0 dB                                                                                                                                                 |
| Follow project selection                             | on                                                                                                                                                                    |
| Expanded sections                                    | Generate, Preview, Voice library and Audio Trimmer. Audio Tools, Generation Parameters, Clone a Voice, Diagnostics, Audio Joiner and Audio Equalizer start collapsed. |



# Script Inventory

Every script this guide is based on, and what it is responsible for. The scene panel (Chapter 4) is not part of this inventory.

## Runtime assembly (Scripts/)

| Script                                  | Responsibility                                                                                                                                    |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| DeepVoiceRuntime.cs                     | The inference engine: loads the ONNX sessions, runs prefill and the two AR loops, manages KV caches, decodes frames, and encodes reference audio. |
| DeepVoiceManifest.cs                    | Wraps <code>runtime_manifest.json</code> .                                                                                                        |
| DeepVoiceProfile.cs                     | The voice profile asset, its thread-safe data copy, and the .npy reader.                                                                          |
| DeepVoicePrompt.cs                      | Text cleaning and packing of the prompt tensor.                                                                                                   |
| DeepVoiceSampler.cs                     | Top-k / top-p / temperature sampling with the Gumbel-max draw and the xoshiro256** generator.                                                     |
| DeepVoiceTokenizer.cs                   | Byte-level BPE tokenizer with a binary cache.                                                                                                     |
| DeepVoiceAudio.cs                       | WAV reading and writing, resampling, padding.                                                                                                     |
| DeepVoiceDsp.cs                         | Trim, join, gain, pitch, EQ biquads, limiter, peaks.                                                                                              |
| MiniJson.cs                             | Minimal JSON parser used for the manifest and tokenizer.                                                                                          |
| AiKodex.DeepVoiceOffline.Runtime.asmdef | Runtime assembly definition; references the ONNX Runtime plugin.                                                                                  |

## Editor assembly (Editor/)

| Script                       | Responsibility                                                                                                     |
|------------------------------|--------------------------------------------------------------------------------------------------------------------|
| DeepVoiceWindow.cs           | Window shell: menu item, layout, header, footer, playback state, background-work scheduler, path helpers.          |
| DeepVoiceWindow.Generate.cs  | Text, Voice Model Settings, Voice Library, Clone a Voice, Diagnostics, Generation Parameters, Output and Generate. |
| DeepVoiceWindow.Preview.cs   | The Preview player.                                                                                                |
| DeepVoiceWindow.Trimmer.cs   | The Audio Trimmer.                                                                                                 |
| DeepVoiceWindow.Joiner.cs    | The Audio Joiner.                                                                                                  |
| DeepVoiceWindow.Equalizer.cs | Volume, pitch, five-band EQ and the response curve.                                                                |
| DeepVoiceWindow.Settings.cs  | What is remembered between sessions, and the reset.                                                                |
| DeepVoiceRuntimeHolder.cs    | Owns the single loaded model; frees it on reload and quit.                                                         |

| Script                                 | Responsibility                                                |
|----------------------------------------|---------------------------------------------------------------|
| DeepVoiceVoiceLibrary.cs               | Finds voice profiles in the project.                          |
| DeepVoiceProfileRepair.cs              | Detects and relinks profiles whose script reference is stale. |
| DeepVoiceModelInspector.cs             | Reads the ONNX graphs and verifies the port's assumptions.    |
| AudioSampleCache.cs                    | Reads AudioClips as mono samples and caches them.             |
| EditorAudioPreview.cs                  | Editor playback through Unity's internal preview API.         |
| WaveformView.cs                        | Waveform rendering, selection overlay and scrubbing.          |
| DeepVoiceStyles.cs                     | Colours, GUI styles and the section themes.                   |
| ReferenceComparer.cs                   | Reference-equality comparer for Unity object dictionaries.    |
| AiKodex.DeepVoiceOffline.Editor.asmdef | Editor-only assembly definition.                              |

## D

APPENDIX

## Glossary

Terms used in this guide.

| Term                                  | Meaning                                                                                                                     |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <b>Autoregressive (AR)</b>            | A model that produces output one step at a time, feeding each result back in as input for the next step.                    |
| <b>Codec</b>                          | A neural audio compressor. Its encoder turns audio into short integer codes; its decoder turns codes back into audio.       |
| <b>Codebook</b>                       | One of ten sets of integer codes that together describe a frame of audio.                                                   |
| <b>Frame</b>                          | One time step of codec output: 2048 samples, about 46.4 ms at 44.1 kHz.                                                     |
| <b>Semantic token</b>                 | The first, coarse code of a frame, predicted by the Slow AR network.                                                        |
| <b>Slow AR / Fast AR</b>              | The two transformers: Slow AR predicts the semantic token for each frame; Fast AR fills in the frame's remaining codebooks. |
| <b>KV cache</b>                       | Stored attention keys and values from earlier positions, reused so each step costs little.                                  |
| <b>Prompt</b>                         | The input to the model: instruction text, reference transcript and codes, and your text.                                    |
| <b>Zero-shot</b>                      | Imitating a voice from a single short reference with no training on that speaker.                                           |
| <b>Reference recording</b>            | The clip a voice is cloned from.                                                                                            |
| <b>Voice profile</b>                  | A project asset holding a reference's codec codes and transcript.                                                           |
| <b>Temperature (Output Variation)</b> | Scales the model's scores before sampling. Lower is steadier, higher is more varied.                                        |
| <b>Top-p (Creativity Range)</b>       | Keeps only the smallest set of candidates whose probabilities add up to p.                                                  |
| <b>Top-k (Choice Variety)</b>         | Keeps only the k most likely candidates.                                                                                    |
| <b>Seed</b>                           | The starting value of the random generator. The same seed reproduces the same draws.                                        |
| <b>ONNX Runtime</b>                   | The engine that executes the model files on your CPU.                                                                       |
| <b>int4 / fp16</b>                    | Numeric precisions of the stored model weights: 4-bit integer and 16-bit float.                                             |
| <b>Biquad / peaking filter</b>        | A two-pole, two-zero filter that boosts or cuts a band of frequencies around a centre.                                      |
| <b>Resampling</b>                     | Converting audio from one sample rate to another.                                                                           |
| <b>Peak limiter</b>                   | Scales audio down so its loudest sample stays under a ceiling.                                                              |