

GNOME Radio (OOP Edition) — Formal Specification

Document: GNOME Radio OOP Specification **Version:** 1.0 (specification), targeting application version 83.x
Status: Draft **Project:** gnome-radio-oop — gnomeradio.org **License:** GNU General Public License v3.0 or later (recommended; see §11)

1. Introduction

1.1 Purpose

This document specifies the architecture, functional behavior, data formats, and external interfaces of **gnome-radio-oop**, an object-oriented rewrite of GNOME Radio implemented in Python with GTK4, libadwaita, PyGObject, and GStreamer. It is intended for contributors, packagers, and integrators, and serves as the normative reference for application behavior from version 83.x onward.

1.2 Scope

gnome-radio-oop is a desktop application for discovering, managing, and playing Internet radio stations on the GNOME platform. The specification covers:

- The desktop application (Python/GTK4/libadwaita edition)

- The station data model and JSON interchange format

- Playback and metadata handling via GStreamer

- Desktop integration via MPRIS2 (D-Bus)

- The interactive map view (libshumate)

- The optional FPGA hardware bridge (iCE40UP5K ICENode)

The standalone HTML5 edition (published at gnomeradio.org/gnome-radio-oop.html) shares the station data format defined in §6 but is otherwise out of scope for this document.

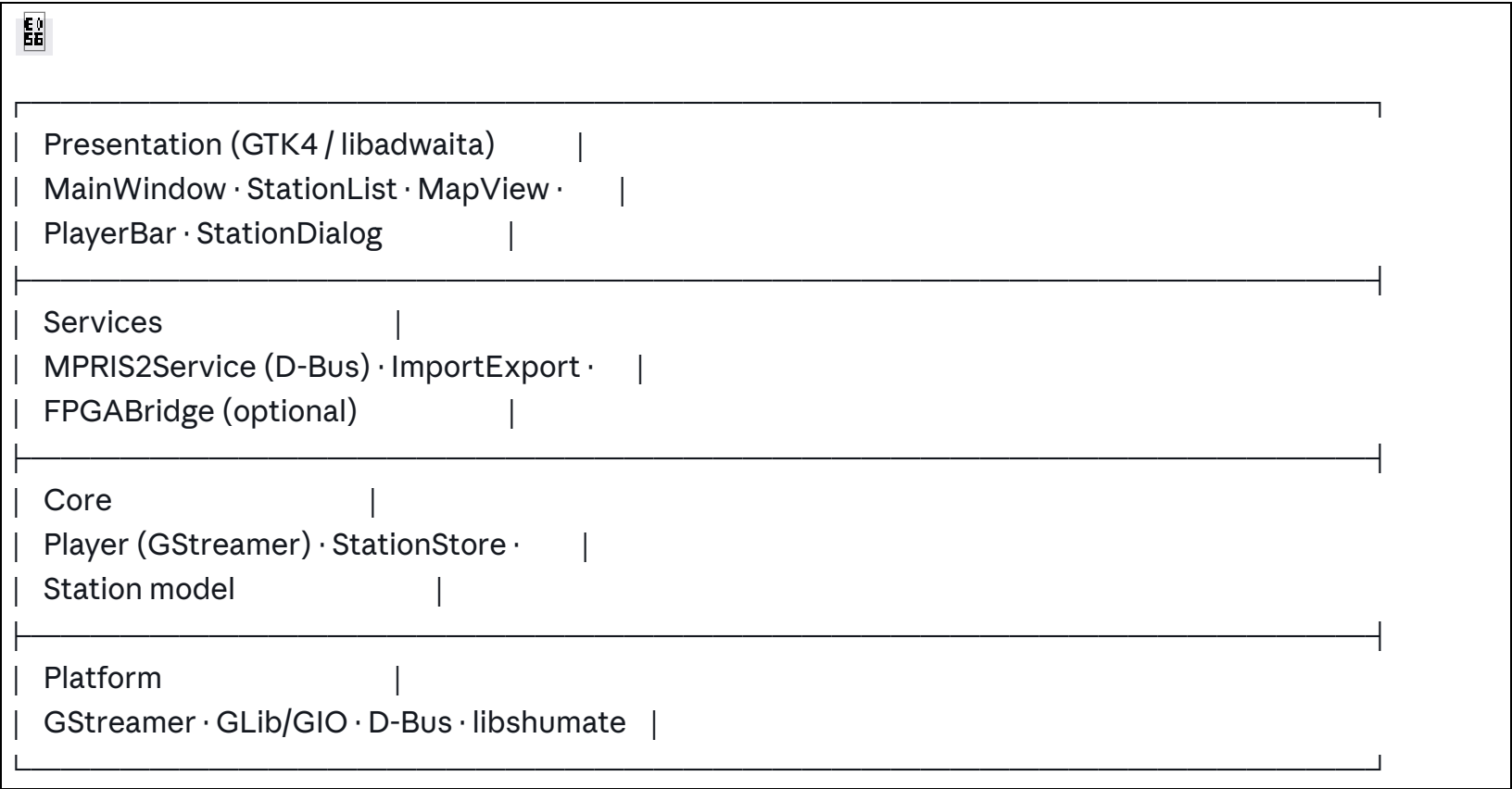
1.3 Definitions and conventions

The key words **MUST**, **MUST NOT**, **SHALL**, **SHOULD**, **SHOULD NOT**, and **MAY** are to be interpreted as described in RFC 2119.

Term	Meaning
Station	A named Internet radio stream with associated metadata
Station library	The persisted collection of stations known to the application
Favorite	A user-flagged station given prominence in the UI
Marker	The visual representation of a station on the map view
ICENode	The iCE40UP5K FPGA board targeted by the hardware bridge

2. System overview

gnome-radio-oop follows a layered object-oriented architecture with a strict separation between the data model, playback engine, desktop integration services, and presentation layer:



Layers MUST depend only downward. The presentation layer MUST NOT access GStreamer directly; all playback control goes through the `Player` object. Services observe the core via signals and MUST NOT hold references into the presentation layer.

3. Runtime environment

3.1 Dependencies

Component	Minimum version	Role
Python	3.10	Implementation language
GTK	4.10	Widget toolkit
libadwaita	1.4	GNOME platform styling and adaptive widgets
PyGObject	3.44	GObject introspection bindings
GStreamer	1.22	Streaming playback (with <code>playbin3</code> , <code>gst-plugins-good</code> , <code>gst-plugins-bad</code> for common stream codecs)
libshumate	1.0	Map rendering
dbus (GLib GDBus)	—	MPRIS2 service

The FPGA bridge additionally requires a serial connection to the ICENode board and SHOULD degrade gracefully to a no-op when no hardware is present (§9).

3.2 Packaging

The application SHALL install with application ID `org.gnome.Radio` (or a namespaced OOP variant, consistently used for the D-Bus name, desktop file, and icon). Reference packaging exists for Arch Linux via the AUR (`gnome-radio`). Distribution packages MUST declare the dependencies in §3.1.

3.3 File locations

Per the XDG Base Directory Specification:

Path	Content
<code>\$XDG_DATA_HOME/gnome-radio/stations.json</code>	Station library (§6)
<code>\$XDG_CONFIG_HOME/gnome-radio/settings</code>	Application settings (GSettings preferred)
<code>\$XDG_CACHE_HOME/gnome-radio/</code>	Map tiles and transient artwork

4. Core model

4.1 Station

`Station` is a GObject subclass with the following properties:

Property	Type	Required	Description
<code>id</code>	str (UUIDv4)	yes	Stable unique identifier
<code>name</code>	str	yes	Display name
<code>url</code>	str	yes	Stream URL (http, https; playlists resolved per §7.2)
<code>homepage</code>	str	no	Station website
<code>genre</code>	str	no	Free-text genre
<code>country</code>	str	no	ISO 3166-1 alpha-2 code
<code>language</code>	str	no	BCP 47 tag
<code>latitude</code>	float	no	WGS84, −90.0 ... 90.0
<code>longitude</code>	float	no	WGS84, −180.0 ... 180.0
<code>favorite</code>	bool	yes (default false)	Favorite flag
<code>created</code>	str (ISO 8601)	yes	Creation timestamp
<code>modified</code>	str (ISO 8601)	yes	Last-modified timestamp

A station is **mappable** if and only if both `latitude` and `longitude` are present and within range. All property mutations MUST emit GObject `notify` signals.

4.2 StationStore

`StationStore` owns the station library and implements `Gio.ListModel` so views bind directly to it.

Requirements:

R-STORE-1 The store SHALL support create, read, update, and delete (CRUD) of stations.

R-STORE-2 Every mutation SHALL be persisted to `stations.json` atomically (write to temporary file, fsync, rename).

R-STORE-3 The store SHALL reject stations whose `url` is empty or whose scheme is not `http` or `https`.

R-STORE-4 `id` collisions on import SHALL be resolved per §8.2.

R-STORE-5 The store SHALL emit `items-changed` on all mutations so that the station list, map view, and MPRIS2 service remain consistent without polling.

R-STORE-6 Toggling `favorite` SHALL be a store-level operation (`set_favorite(id, bool)`) so persistence and signals are uniform.

4.3 Player

`Player` wraps a single GStreamer `playbin3` pipeline.

States: `STOPPED → BUFFERING → PLAYING`, with `ERROR` reachable from any state. Exactly one station may be active at a time.

R-PLAY-1 `play(station)` SHALL stop any current playback before starting the new stream.

R-PLAY-2 The player SHALL expose `volume` (0.0–1.0, cubic scale for UI mapping) and `muted` properties.

R-PLAY-3 Stream tags (see §7.3) SHALL be surfaced as a `metadata-changed` signal carrying title, artist, and station-supplied name when available.

R-PLAY-4 On unrecoverable pipeline errors the player SHALL enter `ERROR`, emit `error(message)`, and release the pipeline; the UI SHALL present the message non-modally (toast).

R-PLAY-5 Network stalls SHALL surface as `BUFFERING` with automatic resume; the player SHOULD retry a dropped stream up to 3 times with exponential backoff before erroring.

5. User interface

The UI is built with libadwaita adaptive widgets and MUST remain usable at widths from 360 px (phone) to desktop sizes.

5.1 Main window

R-UI-1 The main window SHALL provide two primary views — **Stations** (list) and **Map** — switchable via an `AdwViewSwitcher`.

R-UI-2 A persistent **player bar** SHALL show the active station, live metadata (§7.3), a play/stop control, and a volume control whenever a station is active.

R-UI-3 The station list SHALL support search-as-you-type over name, genre, and country, and a filter for favorites.

R-UI-4 Each list row SHALL offer play, favorite toggle, edit, and delete actions; delete SHALL be confirmable or undoable (toast with Undo).

5.2 Station dialog

R-UI-5 Creating and editing stations SHALL use a single `StationDialog` with validation: `name` and `url` required; coordinates validated to range; invalid input blocks saving with inline errors.

5.3 Map view

R-MAP-1 The map view SHALL render all mappable stations as markers using libshumate.

R-MAP-2 Markers SHALL use the Tango palette; the active station's marker SHALL be visually distinct.

R-MAP-3 Activating (clicking/tapping) a marker SHALL start playback of that station (click-to-play) and show a popover with name, genre, and favorite toggle.

R-MAP-4 The map SHALL restore its last center and zoom level across sessions.

R-MAP-5 Tile fetching SHALL be cached (§3.3) and the view SHALL remain functional (grey tiles, working markers) when offline.

5.4 Keyboard and accessibility

R-UI-6 Space SHALL toggle play/stop when the list has focus; `Ctrl+F` focuses search; `Ctrl+N` opens the new-station dialog.

R-UI-7 All interactive elements SHALL carry accessible names; the application SHOULD pass `gtk4-broadwayd`-independent Orca navigation of the station list.

6. Station JSON format

The station library and the import/export interchange format are the same JSON schema, shared with the HTML5 edition.



```
json
{
  "format": "gnome-radio-stations",
  "version": 1,
  "stations": [
    {
      "id": "7f9c3a5e-2f6b-4b9a-9d2e-1c8f0a4b6d21",
      "name": "NRK P2",
      "url": "https://lyd.nrk.no/nrk_radio_p2_mp3_h",
      "homepage": "https://radio.nrk.no/direkte/p2",
      "genre": "Culture",
      "country": "NO",
      "language": "nb",
      "latitude": 59.9139,
      "longitude": 10.7522,
      "favorite": true,
      "created": "2026-01-15T09:00:00Z",
      "modified": "2026-08-01T12:30:00Z"
    }
  ]
}
```

R-JSON-1 Files MUST be UTF-8 encoded, with a top-level `format` of `"gnome-radio-stations"` and integer `version`.

R-JSON-2 Readers MUST ignore unknown keys (forward compatibility) and MUST reject files whose `version` is greater than the highest supported version, with a clear error.

R-JSON-3 Optional fields are omitted rather than set to `null`.

7. Playback and metadata

7.1 Supported streams

The application SHALL play any stream GStreamer's installed plugin set supports; MP3, AAC, and Ogg Vorbis/Opus over HTTP(S), including Icecast/SHOUTcast, are the reference targets.

7.2 Playlist resolution

URLs ending in `.m3u`, `.m3u8` (non-HLS), or `.pls` SHOULD be resolved to their first playable entry before pipeline construction. HLS (`.m3u8`) streams are handed to GStreamer directly.

7.3 Metadata

Tags arriving on the GStreamer bus (`title`, `artist`, `organization`) SHALL update the player bar and the MPRIS2 `Metadata` property within 1 second of receipt. Icecast `StreamTitle` in-band metadata SHALL be honored.

8. Import and export

R-IE-1 Export SHALL write the complete library, or the current filtered selection, in the §6 format via a `Gtk.FileDialog`.

R-IE-2 Import SHALL parse a §6 file and merge it into the library.

R-IE-3 Merge semantics: matching `id` → update existing station if the incoming `modified` is newer; unknown `id` → insert; identical `url` under a different `id` MAY be flagged to the user as a probable duplicate.

R-IE-4 Import SHALL be all-or-nothing per file: a malformed file changes nothing and reports the first error with line context where available.

9. MPRIS2 integration

The application SHALL publish an MPRIS2 service on the session bus.

R-MPRIS-1 Bus name: `org.mpris.MediaPlayer2.GnomeRadio`; object path `/org/mpris/MediaPlayer2`.

R-MPRIS-2 Interfaces implemented: `org.mpris.MediaPlayer2` (with `Identity`, `DesktopEntry`, `Raise`, `Quit`) and `org.mpris.MediaPlayer2.Player`.

R-MPRIS-3 `PlaybackStatus` SHALL map `PLAYING` → `"Playing"`, all other states → `"Stopped"` (live radio has no meaningful pause; `Pause` MAY map to stop, and `CanPause` SHALL be `false`, `CanSeek` `false`).

R-MPRIS-4 `Metadata` SHALL include `mpris:trackid`, `xesam:title` (current track title or station name), and `xesam:artist` when known.

R-MPRIS-5 Property changes SHALL be signalled via `org.freedesktop.DBus.Properties.PropertiesChanged` so GNOME Shell's media controls stay current.

10. FPGA hardware bridge (optional)

The `FPGABridge` service connects the application to the iCE40UP5K **ICENode** board via a Verilog/Python bridge.

R-FPGA-1 The bridge is strictly optional: absence of hardware, serial permissions, or the bridge's Python dependencies MUST NOT affect any other feature.

R-FPGA-2 The bridge SHALL auto-detect the ICENode on well-known serial device paths and expose a connection state (`disconnected`, `connected`, `error`) in the UI (e.g., a status indicator in the primary menu).

R-FPGA-3 Outbound events: playback state changes and now-playing metadata SHALL be forwarded to the board over the serial protocol.

R-FPGA-4 Inbound events: hardware controls (e.g., station step, volume) SHALL map to the same actions as their UI equivalents, going through `Player`/`StationStore` — never manipulating widgets directly.

R-FPGA-5 The serial protocol SHALL be versioned; a version mismatch SHALL yield `error` state with a diagnostic, not silent misbehavior.

11. Non-functional requirements

R-NF-1 Startup: cold start to interactive station list in ≤ 1.5 s on reference hardware with a 500-station library.

R-NF-2 Scale: the list and map SHALL remain responsive with 5,000 stations (list virtualization via `Gtk.ListView`; marker clustering MAY be used above 1,000 mappable stations).

R-NF-3 Robustness: the application SHALL never lose the station library on crash (atomic writes, §4.2) and SHALL start with an empty library plus a warning if `stations.json` is unreadable, preserving the damaged file as `stations.json.bak`.

R-NF-4 Privacy: the application SHALL make no network requests other than stream playback, playlist resolution, and map tiles; no telemetry.

R-NF-5 Internationalization: all user-visible strings SHALL be translatable via gettext; Norwegian Bokmål and English SHALL be maintained reference translations.

R-NF-6 Licensing: the application and this specification's reference implementation SHALL be released under GPL-3.0-or-later, consistent with GNOME ecosystem practice.

12. Versioning and releases

Releases follow GNOME-style versioning (e.g., 80.0, 81.0, 82.0, 83.0.x). Within this specification's lifetime:

Patch releases (83.0.x) MUST NOT change the JSON format version or D-Bus interface.

The JSON `version` field increments only on backward-incompatible schema changes, accompanied by a migration path in the release notes.

13. Testing requirements

T-1 Unit tests for `Station` validation, `StationStore` CRUD/persistence, and JSON round-tripping (export → import yields an equal library).

T-2 An MPRIS2 conformance check driving the service over a private session bus.

T-3 A GStreamer integration test against a local Icecast fixture covering play, metadata tags, and error recovery.

T-4 The FPGA bridge SHALL be testable against a mock serial endpoint; hardware-in-the-loop tests are OPTIONAL.

End of specification.