Niori

Manual

Igor Anikeev

Copyright © 2026 Igor Anikeev.
This manual documents Niori, a medical information manager.

Contents

1 Overview 1
- 1.1 What is Niori . . . 1
- 1.2 Features . . . 1
- 1.3 Requirements . . . 2
- 1.4 Concepts . . . 2

2 Installation 3
- 2.1 Installing as a package . . . 3
- 2.2 Installing as a standalone application . . . 3
- 2.3 First run . . . 4
- 2.4 Upgrading . . . 4

3 Using Niori 5
- 3.1 Starting out . . . 5
- 3.2 The dashboard . . . 6
- 3.3 Rosters . . . 6
- 3.4 Forms . . . 6
- 3.5 Detail views . . . 7
- 3.6 Medications . . . 7
- 3.7 Visits and letters . . . 7
- 3.8 Lab tests . . . 7
- 3.9 Vaccinations . . . 7
- 3.10 Documents . . . 7
- 3.11 Exporting data . . . 8
- 3.12 History and revisions . . . 8
- 3.13 Backups . . . 8
- 3.14 FHIR interoperability . . . 8

4 Configuration 9
- 4.1 Customizing Niori . . . 9
- 4.2 Variable reference . . . 9
- 4.3 Environment variables . . . 10
- 4.4 Appearance . . . 10
- 4.5 Text size . . . 10
- 4.6 Optional integrations . . . 11

5 Data and Privacy 12
- 5.1 The data directory . . . 12
- 5.2 File permissions . . . 12
- 5.3 Revisions and audit trail . . . 13
- 5.4 Backups and relocation . . . 13
- 5.5 Attachment safety . . . 13
- 5.6 Encryption . . . 13
- 5.7 Network access . . . 13
- 5.8 Schema migrations . . . 14

6 Troubleshooting **15**
6.1 No SQLite support . . . 15
6.2 Offline fallback . . . 15
6.3 Permission warnings . . . 15
6.4 Documents will not open . . . 15
6.5 Spell-checking does not start . . . 15
6.6 Encrypted export fails . . . 15
6.7 FHIR retrieval fails . . . 16
6.8 Orphaned attachments . . . 16
6.9 Key binding conflicts . . . 16
6.10 Restoring a database . . . 16

A Introduction to the Emacs Way **17**
A.1 How key bindings are written . . . 17
A.2 A survival kit . . . 17
A.3 Words Emacs uses . . . 18
A.4 GPG and EasyPG . . . 19
A.5 Running without an init file . . . 19

Chapter 1

Overview

This chapter introduces Niori, its purpose and its main concepts.

1.1 What is Niori

Niori is a personal medical information manager for GNU Emacs. It keeps your *medications*, *visits*, *lab tests* and *vaccinations* in a single local SQLite database, presented through a dashboard that surfaces what needs attention: upcoming appointments, abnormal results and boosters that are due. The reports and scans behind those records (PDF, DOCX, images) live in the same data directory, attached to the record they belong to.

Rather than reimplementing UI toolkits, it composes the platform Emacs already provides: `tabulated-list-mode` rosters, `widget` forms, the built-in `sqlite` binding, Modus themes and transient menus.

It runs as a package inside your Emacs, or as a standalone application on a stock, configuration-free Emacs (see Section 2.2 [Installing as a standalone application], page 3). Your data stays yours: one data directory holds the database and every attached document, offline by default, with an optional GPG-encrypted Org export and automatic revision snapshots you can diff with `ediff` (see Section 3.12 [History and revisions], page 8). Niori is free software under GPL-3.0-or-later, uses lexical binding everywhere and depends on GNU ELPA packages only. This manual describes version 0.1.0 of niori.el.

If Emacs is new to you, start with Chapter A [Introduction to the Emacs way], page 17 (Appendix A): it explains how the key bindings in this manual are written, the words Emacs uses for everyday things, and what GPG and EasyPG are. If you are already familiar with Emacs, you can skip this appendix. For more about Emacs itself, see the GNU Emacs web site (`https://www.gnu.org/software/emacs/`).

1.2 Features

Niori tracks four kinds of clinical records. Each is an independent module with its own roster, form and dashboard section:

Module	What you track	Dashboard section
Medications	name, dosage, form, dates, prescriber, instructions	active medications list
Visits	date, doctor, facility, reason, diagnosis, notes	upcoming visits (next 30 days)
Lab tests	date, name, result, unit, reference range, flag	latest result per test, abnormal flags
Vaccinations	date, vaccine, dose, route, lot, provider, next due	boosters due / overdue

Several features cut across all modules:

- *Rosters* (sortable `tabulated-list-mode` buffers), *forms* (validated `widget` buffers with date checks) and *detail views* with buttons.
- *Documents*: attach any file (PDF, DOCX, image, …) to a record; it is copied into the data directory and opened with `doc-view`, `image-mode` or an external viewer (see Section 3.10 [Documents], page 7).
- *Letters* composed from visit records, with `flyspell` spell-checking and optional `yasnippet` templates (see Section 3.7 [Visits and letters], page 7).
- *Exports*: Org (optionally encrypted), CSV and FHIR JSON (see Section 3.11 [Exporting data], page 8).
- *History*: every change or deletion is snapshotted, and any record can be diffed against its previous revision (see Section 3.12 [History and revisions], page 8).

- *Backups*: one keystroke copies the database to a timestamped file (see Section 3.13 [Backups], page 8).
- FHIR R4 export plus an authenticated FHIR client (see Section 3.14 [FHIR interoperability], page 8).

1.3 Requirements

Niori needs GNU Emacs 29.1 or later, built with SQLite support. That is the default on major distributions; Niori checks at startup and tells you if SQLite is missing (see Section 6.1 [No SQLite support], page 15).

The following tools are optional; the features they support degrade gracefully without them:

`gpg`
: Needed for the encrypted Org export (see Section 5.6 [Encryption], page 13).

`hunspell` or `aspell`
: Needed for spell-checking letters (see Section 3.7 [Visits and letters], page 7).

A system opener
: `xdg-open` on GNU/Linux, `open` on macOS, the shell's own associations on MS-Windows, for documents Emacs cannot display (DOCX, XLSX, …) (see Section 3.10 [Documents], page 7).

A FHIR server
: Only if you want to retrieve records from one (see Section 3.14 [FHIR interoperability], page 8).

The GNU ELPA dependencies `compat`, `transient`, `auto-dark` and `fontaine` are installed automatically when you install the package (see Section 2.1 [Installing as a package], page 3).

1.4 Concepts

These terms recur throughout the manual:

Module
: A clinical domain (medications, visits, lab tests, vaccinations). A module registers its database schema, roster columns, form fields and dashboard section; a generic view layer renders all of it, so a module file contains no buffer code of its own.

Roster
: A sortable list buffer showing all records of one kind (see Section 3.3 [Rosters], page 6).

Form
: A `widget` buffer for creating or editing one record, with validated fields and date checks (see Section 3.4 [Forms], page 6).

Detail view
: A read-only buffer describing one record, with buttons for the actions that apply to it (see Section 3.5 [Detail views], page 7).

Dashboard
: The central buffer combining each module's section into one overview of what needs attention (see Section 3.2 [The dashboard], page 6).

Data directory
: The single directory holding the SQLite database, attached documents, backups and exports; the unit of backup, sync and relocation (see Section 5.1 [The data directory], page 12).

Revision
: A JSON snapshot of the previous state of a record, written automatically on every update or delete (see Section 5.3 [Revisions and audit trail], page 13).

Chapter 2

Installation

This chapter describes how to obtain, install and run Niori.

2.1 Installing as a package

The package is called `niori` (this repository is niori.el (`https://github.com/ianikeev/niori.el`)). Once it is on MELPA, install it with `use-package`:

```
(use-package niori
  :ensure t
  :commands (niori niori-dashboard))
```

To install it straight from a checkout while it is not yet packaged, add the repository to your load path:

```
(use-package niori
  :load-path "~/src/niori.el"
  :commands (niori niori-dashboard))
```

The GNU ELPA dependencies (`compat`, `transient`, `auto-dark`, `fontaine`) are declared in `niori.el` and installed automatically; `package.el` users who prefer `use-package` need nothing else. A MELPA recipe ships in `recipes/niori`.

2.2 Installing as a standalone application

Niori can run as a full application on a stock, configuration-free Emacs (`emacs -Q`; see Section A.5 [Running without an init file], page 19):

```
git clone https://github.com/ianikeev/niori.el
./niori.el/bin/niori
```

That is all. On the first run (when online) the launcher installs the GNU ELPA dependencies plus `vertico`, `consult`, `orderless` and `which-key`; run fully offline it degrades gracefully to built-ins (Modus themes with day/night switching, `fido-vertical-mode`, plain keymap help). Set the environment variable `NIORI_NO_NET` to `1` to always start offline (see Section 4.3 [Environment variables], page 10).

To install it properly, with a desktop entry:

```
make install             # defaults to ~/.local
# or: make install PREFIX=/usr/local
```

This installs `bin/niori`, the Elisp sources under `share/emacs/site-lisp/niori/`, the icon, and `etc/niori.desktop` rewritten with the absolute `Exec=` path.

2.3 First run

The first time Niori opens its database, it creates the data directory (owner-only, mode `0700`, with the `.niori-data` layout marker) and the SQLite file inside it, then runs any pending migrations (see Section 5.8 [Schema migrations], page 14). The database starts empty: there is no seed data. Open the dashboard with `M-x niori` and add your first record (see Section 3.1 [Starting out], page 5). If Emacs is new to you, read the survival kit in Section A.2 [A survival kit], page 17 first.

2.4 Upgrading

Schema changes run through numbered migrations, recorded in the `schema_migrations` table. A new version upgrades your database in place: you install it and keep using the same data directory. New *tables*—like the `attachments` table of a later release—appear on the next open without a migration.

Chapter 3

Using Niori

This chapter explains the application's features and workflows.

3.1 Starting out

Type `M-x niori` to open the dashboard and enable the key bindings. The command turns on `niori-mode`, a global minor mode, so the `C-c M` prefix works in every buffer; until then Niori has zero footprint in your Emacs. Everything is reachable from the dashboard's buttons, or from the keyboard:

`C-c M d`
: Open the dashboard (see Section 3.2 [The dashboard], page 6).

`C-c M m`
: Medications roster (see Section 3.6 [Medications], page 7).

`C-c M v`
: Visits roster (see Section 3.7 [Visits and letters], page 7).

`C-c M t`
: Lab tests roster (see Section 3.8 [Lab tests], page 7).

`C-c M c`
: Vaccinations roster (see Section 3.9 [Vaccinations], page 7).

`C-c M a` *module*
: Start a new record: `C-c M a m` for a medication, and likewise for visits, lab tests and vaccinations (see Section 3.4 [Forms], page 6).

`C-c M E o`
`C-c M E c`
`C-c M E j`
: Export a module as Org, CSV or FHIR JSON (see Section 3.11 [Exporting data], page 8).

`C-c M D`
: Diff a record against its last revision (see Section 3.12 [History and revisions], page 8).

`C-c M f`
: Show every document you have attached (see Section 3.10 [Documents], page 7).

`C-c M l`
: Compose a letter from a visit (see Section 3.7 [Visits and letters], page 7).

`C-c M b`
: Back up the database (see Section 3.13 [Backups], page 8).

`C-c M ?`
: Open the transient command menu (when `transient` is installed).

`C-c M C-h`
: Show a full overview of every Niori key binding.

CUA bindings (C-x/C-c/C-v cut/copy/paste) are available for standalone users: set niori-standalone-cua to t, or type M-x cua-mode anytime. Text size follows the desktop zoom keys C-=, C-+ and C-- (see Section 4.5 [Text size], page 10).

3.2 The dashboard

The dashboard is Niori's home screen; M-x niori-dashboard also opens it directly. Each module contributes a section, so one buffer shows, at a glance:

- your active medications;
- upcoming visits, within niori-visits-upcoming-days days (default 30);
- the latest result of each lab test, with abnormal results flagged;
- vaccinations whose booster is due or overdue, within niori-vaccinations-due-days days (default 90).

Entries are buttons: follow one to open the record's detail view. After every write, Niori refreshes any open dashboard, roster and detail buffers for the affected module, so what you see is always current.

3.3 Rosters

A roster lists all records of one kind in a sortable tabulated-list-mode buffer. Open one from the dashboard or with C-c M m (medications), C-c M v (visits), C-c M t (lab tests) or C-c M c (vaccinations).

RET
: Open the record under point in its detail view.

a
: Add a new record (the same form as C-c M a …).

e
: Edit the record under point.

d
: Delete the record under point; Niori asks first (see Section 5.5 [Attachment safety], page 13).

g
: Refresh the roster.

S
: Sort; clicking a column header sorts as well.

D
: Diff the record under point against its previous revision (see Section 3.12 [History and revisions], page 8).

E
: Export the module (see Section 3.11 [Exporting data], page 8).

L
: Compose a letter from the visit under point (visits roster only, see Section 3.7 [Visits and letters], page 7).

3.4 Forms

Records are created and edited in widget forms: buffers with labelled, validated fields, including checks on dates, so invalid data never reaches the database.

C-c C-c
: Save the record.

C-c C-k
: Cancel and leave the form.

3.5 Detail views

The detail view describes one record read-only, with buttons for the actions that apply to it. It also accepts the roster keys: `e` edits, `d` deletes, `D` diffs against the previous revision, `L` composes a letter, and `E` opens the export menu. In every detail view, `A` attaches a document to the record (see Section 3.10 [Documents], page 7). After a write, all open buffers of that module refresh automatically.

3.6 Medications

The medications module tracks name, dosage, form, dates, prescriber and instructions for each medication. The dashboard lists your active medications. Open the roster with `C-c M m`; start a new record with `C-c M a m`, or with `a` from the roster.

3.7 Visits and letters

The visits module tracks date, doctor, facility, reason, diagnosis and notes for each visit; the dashboard lists upcoming visits within the configured window (see Section 4.2 [Variable reference], page 9). In the visits roster, `L` (or `C-c M l`) composes a *letter* from the record under point: a narrative document describing the visit, which you edit and file as you would any other document. Letters are spell-checked with `flyspell` when a `hunspell` or `aspell` binary exists (`niori-letter-flyspell`), and optional `yasnippet` templates ship in `snippets/` (see Section 4.6 [Optional integrations], page 11).

3.8 Lab tests

The lab tests module tracks date, name, result, unit, reference range and flag. The dashboard shows the latest result of each test, with abnormal flags, so an out-of-range result surfaces even when it was recorded weeks ago. Open the roster with `C-c M t`.

3.9 Vaccinations

The vaccinations module tracks date, vaccine, dose, route, lot, provider and next due. The dashboard lists boosters that are due or overdue within the configured window. Open the roster with `C-c M c`.

3.10 Documents

Any file—a PDF lab report, a DOCX letter, a photograph—can be attached to a visit, lab test, medication or vaccination. Attach it with `A` in the record's detail view. The file is copied into the data directory, one folder per record, so copying or syncing the data directory backs up the whole record (see Section 5.1 [The data directory], page 12); `niori-attachments-import-method` can also move or symlink instead (see Section 4.2 [Variable reference], page 9).

Attached documents open with `doc-view` (PDF), `image-mode` (images) or an external viewer for formats Emacs cannot display, such as DOCX or XLSX.

`C-c M f` opens the list of every document you have attached:

`RET`
: Open the document.

`f`
: Open it with the external viewer.

`o`
: Show its folder in Dired.

`r`
: Retitle it.

`d`
: Detach it; Niori asks before removing the file.

The safety rules behind these buffers—relative paths, checksums and orphan handling—are described in Section 5.5 [Attachment safety], page 13.

3.11 Exporting data

Three export formats cover the common needs, and the output lands in `exports/` inside the data directory (see Section 5.1 [The data directory], page 12):

`C-c M E o`
: Export as Org (`niori-export-org`).

`C-c M E c`
: Export as CSV.

`C-c M E j`
: Export as FHIR JSON (see Section 3.14 [FHIR interoperability], page 8).

The Org export can be encrypted with EasyPG: invoke it with a prefix argument, `C-u C-c M E o` (see Section 5.6 [Encryption], page 13). CSV exports open in `csv-mode` automatically when that package is installed. If you keep narrative notes in Org files of your own, `org-ql` can run structured queries over exported Org archives (see Section 4.6 [Optional integrations], page 11).

3.12 History and revisions

Every update and delete first snapshots the current record, as JSON, into the `revisions` table—an audit trail inside the same database file (see Section 5.3 [Revisions and audit trail], page 13). Whether revisions are kept at all is controlled by `niori-store-keep-revisions` (see Section 4.2 [Variable reference], page 9).

`D` on any record in a roster, or `C-c M D`, opens an `ediff` between the pretty-printed current record and its previous revision, so you can see exactly what changed.

3.13 Backups

`C-c M b` (`M-x niori-store-backup`) copies the database to a timestamped file in `backups/` inside the data directory. Keep a copy on offline media; restoring one is described in Section 6.10 [Restoring a database], page 16. Because the data directory also holds every attached document, the directory itself is the unit of backup and relocation (see Section 5.4 [Backups and relocation], page 13).

3.14 FHIR interoperability

`C-c M E j` (`M-x niori-fhir-export`) writes a module as a FHIR R4 collection Bundle:

Medications
: become `MedicationStatement` resources;

Visits
: become `Encounter` resources;

Lab tests
: become `Observation` resources (numeric results as `valueQuantity`, flags as `H`, `L` and `A` interpretation codes);

Vaccinations
: become `Immunization` resources.

The mapping is intentionally minimal—display strings, no code-system translation—which is right for personal export and review.

For retrieval, set `niori-fhir-server-url` to the server's base URL and keep the bearer token in `auth-source`; then `M-x niori-fhir-fetch` issues authenticated requests with the built-in `url` library and returns parsed JSON (see Section 5.7 [Network access], page 13).

Chapter 4

Configuration

This chapter documents application settings and configuration files.

4.1 Customizing Niori

All user options live in the `niori` customization group: type `M-x customize-group RET niori RET` to browse them with descriptions and types, or set them with `setq` in your init file. There is no configuration file of Niori's own to edit. The standalone application runs without an init file; it is configured through its built-in defaults and the environment variables (see Section 4.3 [Environment variables], page 10).

When Niori runs inside your Emacs it never touches your theme or your fonts; the appearance layer only activates for the standalone application (see Section 4.4 [Appearance], page 10).

4.2 Variable reference

The essentials, all in the `niori` group:

`niori-data-directory`
: The data directory: database, documents, backups, exports. Default `~/.local/share/niori`, subject to the environment (see Section 4.3 [Environment variables], page 10).

`niori-data-file`
: Default `nil`, meaning the database lives inside the data directory. An absolute path here puts the database elsewhere.

`niori-documents-directory`
: Default `nil`, meaning attached documents live inside the data directory. An absolute path here puts them elsewhere.

`niori-attachments-import-method`
: How an attached file enters the data directory: `copy` (default), `move` or `symlink`.

`niori-attachments-external-opener`
: External viewer for formats Emacs cannot display: `xdg-open` on GNU/Linux, `open` on macOS; MS-Windows uses `w32-shell-execute`.

`niori-store-tighten-data-directory`
: What happens on open when other users can reach the data directory: `t` (default) tightens it to owner-only, `advise` keeps your modes but reports once per session, `nil` stays silent (see Section 5.2 [File permissions], page 12).

`niori-store-keep-revisions`
: Keep the revision audit trail; default `t` (see Section 5.3 [Revisions and audit trail], page 13).

`niori-visits-upcoming-days`
: Dashboard window for upcoming visits; default 30.

`niori-vaccinations-due-days`
: Dashboard window for boosters due; default 90.

`niori-letter-flyspell`
: Whether letters are spell-checked; default `auto`, which enables checking when a spell-checking binary exists (see Section 3.7 [Visits and letters], page 7).

`niori-encryption-recipients`
: GPG keys for encrypted exports; default `nil` (see Section 5.6 [Encryption], page 13).

`niori-fhir-server-url`
: Base URL of a FHIR server for retrieval; default `nil` (see Section 3.14 [FHIR interoperability], page 8).

`niori-appearance-manage`
: Whether Niori manages theme and font; default `nil`, `t` in the standalone application (see Section 4.4 [Appearance], page 10).

`niori-appearance-font-family`
: Monospaced font used when the appearance layer manages fonts; auto-detected by default.

`niori-standalone-cua`
: Set to `t` to give standalone users the usual CUA cut/copy/paste bindings (see Section 3.1 [Starting out], page 5).

4.3 Environment variables

Two environment variables matter most:

`NIORI_DATA_DIR`
: Overrides `niori-data-directory`. Useful for the standalone application, which has no init file to customize.

`NIORI_NO_NET`
: Set to `1` to make the standalone bootstrap skip all package installation and always start offline.

The data directory itself resolves in this order: `$NIORI_DATA_DIR` when set; otherwise `$XDG_DATA_HOME/niori` (usually `~/.local/share/niori`); on MS-Windows, `%LOCALAPPDATA%\niori`.

4.4 Appearance

Inside your Emacs, Niori never touches your theme or your fonts: `niori-appearance-manage` is `nil` by default and the appearance layer stays out of the way. It only activates for the standalone application, which presents a complete, coherent look on its own. There it prefers the GNU ELPA `auto-dark` package, switching between the `modus-operandi` and `modus-vivendi` themes with the day/night cycle, and `fontaine` for font presets, with built-in fallbacks when those packages are not available. The font family is auto-detected; set `niori-appearance-font-family` to override it.

4.5 Text size

Once `niori-mode` is enabled (`M-x niori`, see Section 3.1 [Starting out], page 5), the application follows the usual desktop zoom keys: `C-=` and `C-+` make the text larger, `C--` makes it smaller. Repeat a key to keep growing or shrinking. The keys work in every buffer, because `niori-mode` is a global minor mode; the scale itself is a per-buffer setting, so a roster, the dashboard and a letter each keep their own size.

`C-=`
`C-+`
: Increase the text size of the current buffer.

`C--`
: Decrease the text size of the current buffer.

`C-x C-0 0`
: Reset the text size of the current buffer to the default (`text-scale-adjust`, built into Emacs).

Note that `C--` shadows the usual negative prefix argument while `niori-mode` is enabled; use `M--` when a command needs a negative argument.

4.6 Optional integrations

Niori prefers mature Emacs components over new code, and every optional integration is guarded at runtime: absence is a graceful downgrade, not an error. When installed, the following light up automatically:

- `transient`: the `C-c M ?` command menu, runtime-defined, with a fallback when absent;
- `which-key`: keymap discoverability (built into Emacs 30, GNU ELPA otherwise);
- `vertico`, `consult`, `orderless`: completion in the standalone application, auto-installed when online, with `fido-vertical-mode` as the fallback;
- `yasnippet`: letter templates from `snippets/`;
- `csv-mode`: CSV exports open in it automatically;
- `org-ql`: structured queries over exported Org archives.

MELPA-only packages are never hard requirements: the package header lists GNU ELPA dependencies only (see Section 1.3 [Requirements], page 2).

Chapter 5

Data and Privacy

This chapter describes how Niori stores and handles information.

5.1 The data directory

Everything Niori writes lives in one directory you choose—the unit of backup, sync and relocation. Its location resolves to `$NIORI_DATA_DIR` when set, otherwise to `$XDG_DATA_HOME/niori` (usually `~/.local/share/niori`), or `%LOCALAPPDATA%\niori` on MS-Windows (see Section 4.3 [Environment variables], page 10):

```
~/.local/share/niori/
|-- .niori-data          layout marker ("layout-version: 1")
|-- niori.db             the database
|-- documents/           attached files, one folder per record
|   `-- visits/42/2026-11-20-lab-report.pdf
|-- backups/             timestamped database copies
|-- exports/             Org / CSV / FHIR output
`-- letters/             reserved for saved letters
```

The `attachments` table stores paths *relative* to the documents directory (`visits/42/2026-11-20-lab-report.pdf`), plus title, size, MIME type and SHA-256 checksum (see Section 5.5 [Attachment safety], page 13).

5.2 File permissions

On POSIX systems everything Niori writes is owner-only: the directory itself (mode `0700`, re-applied on every open—you are told when an existing loose directory is tightened), the documents inside it, the database, the backups and exports, and the `-journal`, `-wal` and `-shm` files SQLite keeps beside the database. Those sidecar files are created while the process umask is narrowed, so they are private from the instant they exist, instead of needing a `chmod` that would come too late.

On MS-Windows none of that means anything: Emacs maps `chmod` to the read-only attribute there, so Niori performs no mode changes at all and leaves protection to the NTFS permissions of the directory—which is why the default is the per-user `%LOCALAPPDATA%` folder rather than `~/.local`.

If you share the directory on purpose, `niori-store-tighten-data-directory` decides what happens:

`t` (default)
: re-apply mode `0700` on open and say so in one line;

`advise`
: keep your modes, report once per session and name the fix;

`nil`
: keep your modes, say nothing.

In the latter two cases, `M-x niori-store-secure-data-directory` fixes the tree on demand, regardless of the policy. It never follows symbolic links, so a document imported with the opt-in `symlink` method keeps the permissions of the file you own elsewhere. A deliberately shared folder—a family NAS, a group-readable backup volume—can therefore stay shared without being silently overridden on every open.

Write-ahead logging is deliberately not enabled: the data directory is meant to be synced as a whole, and a live `-wal` file on a Syncthing or network folder is a corruption source rather than a performance win.

5.3 Revisions and audit trail

Medical data is edited rarely, but mistakes matter. Every update or delete first snapshots the previous record, as JSON, into the `revisions` table: a real audit trail inside the same database file, at the cost of one JSON write per change. The behaviour is controlled by `niori-store-keep-revisions` (default `t`).

To inspect a change, `D` in a roster or `C-c M D` diffs the record against its previous revision with `ediff` (see Section 3.12 [History and revisions], page 8).

5.4 Backups and relocation

`C-c M b` writes a timestamped copy of the database into `backups/` (see Section 3.13 [Backups], page 8); keep a copy on offline media. Because attaching copies files by default, the data directory is a complete copy of the record, and "back up this folder" is sufficient advice.

Moving an installation is a directory move plus `M-x niori-store-switch-data-directory`: the command closes the connection, re-points `niori-data-directory`, reopens the database (running migrations) and refreshes open buffers. It asks before adopting a folder that holds no Niori data yet, and keeps using the old directory if the new one cannot be opened.

5.5 Attachment safety

Documents are treated with care:

- Paths are stored relative to the documents directory, and expanding a stored path refuses anything that would escape it, so a hand-edited row cannot turn into an arbitrary file read.
- Attaching *copies* the file by default (the original stays where it was); `niori-attachments-import-method` allows `move` and `symlink` for large scans, at the cost of no longer being self-contained.
- Detaching asks before removing a file, and never removes one another row still points at.
- Deleting a *record* asks first and keeps the files. `M-x niori-attachments-check` lists missing files and documents orphaned by a deleted record; `M-x niori-attachments-purge-orphans` cleans the orphans up.

5.6 Encryption

The Org export can be encrypted with EasyPG: invoke it with a prefix argument, `C-u C-c M E o`, and it is encrypted to the keys in `niori-encryption-recipients`. The gpg prerequisite is listed in Section 1.3 [Requirements], page 2; what GPG and EasyPG actually are is explained in Section A.4 [GPG and EasyPG], page 19, written for readers meeting them for the first time.

If you keep narrative notes in Org files instead, the built-in `org-crypt` encrypts subtrees on save—Niori stays out of the way of both styles.

5.7 Network access

Nothing leaves your machine unless you call the FHIR client. There is no telemetry and no sync service; the data directory is offline by default.

For FHIR retrieval, set `niori-fhir-server-url` to the server's base URL and keep the bearer token in `auth-source`—a line like the following in `.authinfo`:

```
machine fhir.example.com secret TOKEN
```

Then `M-x niori-fhir-fetch` issues authenticated requests with the built-in `url` library and returns parsed JSON (see Section 3.14 [FHIR interoperability], page 8).

5.8 Schema migrations

Schema changes run through numbered migrations recorded in the `schema_migrations` table. A new version upgrades the database in place; new tables appear on the next open without a migration (see Section 2.4 [Upgrading], page 4). Table and column names come exclusively from the schema registry, and values are always bound parameters, so the query surface is injection-safe.

Chapter 6

Troubleshooting

This chapter covers common problems and their solutions.

6.1 No SQLite support

Niori needs Emacs 29.1 or later built with SQLite support. That is the default on major distributions, and Niori checks at startup and tells you if the binding is missing. Install an Emacs build with SQLite support (see Section 1.3 [Requirements], page 2) and Niori will open its database.

6.2 Offline fallback

If the standalone application starts with plain themes and simple completion, it could not reach GNU ELPA. This is deliberate: every optional package is guarded, and the application must start on a bare, offline Emacs. The first run installs `vertico`, `consult`, `orderless` and `which-key` when online; set `NIORI_NO_NET` to `1` to always start offline (see Section 4.3 [Environment variables], page 10). On a degraded start you still get Modus themes with day/night switching, `fido-vertical-mode` and plain keymap help.

6.3 Permission warnings

If Niori reports a loose data directory on open, either let it tighten the directory (the default, `niori-store-tighten-data-directo t`), or, if you share the directory on purpose, set the option to `advise` or `nil` and run `M-x niori-store-secure-data-directory` whenever you want it fixed (see Section 5.2 [File permissions], page 12).

6.4 Documents will not open

PDFs open with `doc-view` and images with `image-mode`; other formats (DOCX, XLSX, …) go to the external opener: the program in `niori-attachments-external-opener`, which defaults to `xdg-open` on GNU/Linux, `open` on macOS, and `w32-shell-execute` on MS-Windows. If nothing happens, check that the opener matches your system (see Section 4.2 [Variable reference], page 9).

6.5 Spell-checking does not start

Letters are spell-checked with `flyspell` only when a `hunspell` or `aspell` binary exists; `niori-letter-flyspell` defaults to `auto`, which enables checking only in that case (see Section 3.7 [Visits and letters], page 7). Install either binary to get spell-checking.

6.6 Encrypted export fails

Encrypted Org export needs two things: `gpg` installed, and at least one key in `niori-encryption-recipients`. Remember that encryption is requested with a prefix argument (`C-u C-c M E o`); without it, the export is written in plain Org (see Section 5.6 [Encryption], page 13).

6.7 FHIR retrieval fails

`M-x niori-fhir-fetch` needs `niori-fhir-server-url` and a bearer token in `auth-source` for the server's host: a `.authinfo` line such as `machine fhir.example.com secret TOKEN` (see Section 5.7 [Network access], page 13).

6.8 Orphaned attachments

Deleting a record keeps its files on purpose (see Section 5.5 [Attachment safety], page 13), so documents can outlive their record. `M-x niori-attachments-check` lists missing files and orphans; `M-x niori-attachments-purge-orphans` removes the orphans. Files still referenced by another record are never removed.

6.9 Key binding conflicts

The `C-c M` prefix was chosen to avoid collisions with other packages that use popular prefixes (for instance `C-c n`, used by deft). `niori-mode` is a global minor mode with zero footprint until you run `M-x niori`. If another package shadows one of the bindings, `C-c M C-h` shows the full keymap overview (see Section 3.1 [Starting out], page 5).

6.10 Restoring a database

`C-c M b` writes timestamped copies into `backups/` (see Section 3.13 [Backups], page 8). To restore one, close the database first with `M-x niori-store-close`, copy the backup over `niori.db` in the data directory, and start Niori again.

Appendix A

Introduction to the Emacs Way

This appendix is for readers who have never used Emacs—perhaps you are coming to Niori as a standalone application, on a computer where Emacs was installed mainly to run it (see Section 2.2 [Installing as a standalone application], page 3). It explains how the key bindings in this manual are written, the small set of keys that gets you out of trouble, the words Emacs uses for everyday things, and what GPG and EasyPG are. If you have used Emacs before, skip this appendix.

A.1 How key bindings are written

Commands in this manual are written as key sequences. `C-` means the *Control* key: `C-x` is "hold Control and type x". `M-` means *Meta*: on most keyboards that is the Alt key, and on keyboards without one, pressing and releasing Esc before the key works as a substitute. `M-x` is therefore "hold Alt and type x".

A sequence of several elements is typed in order, one after another. `C-c M b`, Niori's backup command, means: hold Control and type c, release everything, then hold Alt and type b. Within one element the keys are held together; between elements they are not.

Letters matter exactly as written: `C-a` and `C-A` are different bindings, and `C-A` means Control with a *shifted* A. A few keys have names instead of symbols: `RET` is the Enter key, `SPC` the space bar, `TAB` the Tab key. Where this manual writes `M-x customize-group RET niori RET`, you type `M-x`, the word `customize-group`, press `RET`, type `niori` and press `RET` again.

`M-x` deserves a second look, because every Niori command can be reached through it: `M-x` asks for a command name in the minibuffer (see Section A.3 [Words Emacs uses], page 18), and `RET` runs it. A prefix argument modifies the command that follows: `C-u C-c M E o` asks the Org export to encrypt itself (see Section 3.11 [Exporting data], page 8).

Emacs documents itself. `C-h k` asks "what does this key do?", `C-h m` describes the current buffer and lists every key binding active in it, and `C-h t` starts the built-in tutorial. Inside Niori, `C-c M C-h` shows all Niori bindings at once (see Section 3.1 [Starting out], page 5).

A.2 A survival kit

Ten keys answer almost every first-day question. None of them are Niori's; they belong to Emacs everywhere.

`C-g`
: Quit whatever is happening: a half-typed key sequence, a minibuffer prompt, a search, a hung command. When in doubt, type `C-g`.

`C-x C-c`
: Leave Emacs.

`C-x C-f`
: Open a file: Emacs asks for its name in the minibuffer. Use this to look at an exported Org or CSV file.

`C-x C-s`
: Save the current buffer to its file.

`C-/`
: Undo the last change (`C-x u` also works).

`C-s`
: Search forward in the buffer; `C-r` searches backwards. `RET` stops the search where you are.

`C-v`
: Scroll down a screenful; `M-v` scrolls up. The `PgDn` and `PgUp` keys do the same.

`M-x`
: Run any command by name (see Section A.1 [How key bindings are written], page 17). Every `niori-` command mentioned in this manual works here, including those without a key binding.

`C-h k`
: Ask what a key does: type the key after `C-h k` and Emacs shows its documentation.

`C-h t`
: The Emacs tutorial: half an hour that repays itself for the rest of the week.

A.3 Words Emacs uses

Emacs calls everyday things by names of its own. These recur throughout the manual:

Buffer
: The content you edit or read—a file, a roster, a form, the dashboard. Niori's rosters, forms and detail views are buffers.

Window
: A pane showing one buffer. Emacs's *frame* is what your desktop calls a window; a *window* in Emacs is one pane of it, and Emacs shows several buffers side by side by splitting the frame into windows.

Point
: The cursor. Commands act at point: in a roster, `RET` opens the record under point (see Section 3.3 [Rosters], page 6).

Minibuffer
: The prompt area at the bottom of the frame, where Emacs asks questions and accepts command names after `M-x`. `RET` confirms and `C-g` cancels. One-line messages appear next to it, in the *echo area*.

Mode
: The active behaviour of a buffer. A *major mode* says what the buffer is—a `tabulated-list-mode` roster, `doc-view` for a PDF—and *minor modes* layer extra behaviour on top. `niori-mode` is a *global* minor mode, active in every buffer, which is why `C-c M` works everywhere (see Section 3.1 [Starting out], page 5).

Keymap
: The table that connects a key to a command; one connection is a *key binding*. Prefix keys like `C-c M` select a whole sub-table: the `b` in `C-c M b` is looked up inside Niori's own keymap.

Init file
: The personal startup script Emacs runs at launch (`~/.emacs.d/init.el` or `~/.emacs`), where `setq` and `use-package` lines go. The standalone application runs *without* one (see Section A.5 [Running without an init file], page 19).

Customize
: The built-in settings browser: `M-x customize-group RET niori RET` lists every Niori option with descriptions and types (see Section 4.1 [Customizing Niori], page 9).

Package
: An installable extension. GNU ELPA is the GNU archive Emacs installs from by default; MELPA is a popular community archive. Niori depends on GNU ELPA packages only (see Section 1.3 [Requirements], page 2).

Completion
: While you type in the minibuffer, Emacs offers matching choices and `RET` picks one. The standalone application installs `vertico`, `consult` and `orderless` for this (see Section 4.6 [Optional integrations], page 11).

A.4 GPG and EasyPG

Two names appear wherever Niori encrypts something (see Section 5.6 [Encryption], page 13).

GPG, the GNU Privacy Guard, is the free OpenPGP implementation: a command-line program, gpg, that encrypts data to key pairs. A *key pair* has a public half, which you may publish and which others encrypt *to* you, and a secret half, protected by a *passphrase*, which decrypts. You generate a pair with `gpg --full-generate-key` and list the ones you have with `gpg --list-keys`; each is identified by a name and an e-mail address. The `niori-encryption-recipients` option holds such identifiers: an encrypted export is encrypted to those public keys, so only someone holding the matching secret keys—you, or a person you chose—can read it. A small helper program, `gpg-agent`, remembers the passphrase for a while, so you need not retype it.

EasyPG is the Emacs side of the same machinery, built into Emacs. The EasyPG Assistant manages keys interactively (`M-x epa-list-keys`), and `epa-file` makes Emacs treat any file whose name ends in `.gpg` transparently: it decrypts on open, re-encrypts on save, and asks for the passphrase in the minibuffer when `gpg-agent` does not still have it. Niori's encrypted export simply hands the file to this layer: open the export in Emacs and you are reading it; `gpg --decrypt` does the same outside Emacs.

A.5 Running without an init file

Emacs normally runs your init file at startup: themes, fonts, key tweaks and installed packages—everything you have put there. The `-Q` flag, short for "quick", starts Emacs without it: no personal init file, no site-wide one, no X resources. You get a pristine, stock editor session that behaves the same on every machine and every run.

That is exactly how Niori's standalone launcher works (see Section 2.2 [Installing as a standalone application], page 3): `bin/niori` starts such a stock Emacs, loads Niori into it and lets you open the dashboard with `M-x niori`, so the application never inherits—and never touches—any personal configuration. Two practical consequences:

- What persists between runs is your data—the data directory (see Section 5.1 [The data directory], page 12)—not settings. To point the standalone application at a particular data directory, or to keep it offline, set the environment variables `NIORI_DATA_DIR` and `NIORI_NO_NET` (see Section 4.3 [Environment variables], page 10); there is no init file to set options in.
- Options you change interactively—`M-x customize-group RET niori RET`, `M-x cua-mode` for cut/copy/paste keys (see Section 3.1 [Starting out], page 5)—last for the session only. If you later move Niori into your own Emacs, the same settings belong in the init file instead (see Section 4.1 [Customizing Niori], page 9).

If a first contact with Emacs itself feels rough, `C-h t` (the tutorial) and the survival kit above are the shortest way in.

Index

Key Index

C-+, 10
C--, 10
C-=, 10
C-c C-c, 6
C-c C-k, 6
C-c M, 5
C-c M ?, 5
C-c M a, 5
C-c M b, 5
C-c M c, 5
C-c M C-h, 5
C-c M D, 5
C-c M d, 5
C-c M E, 5
C-c M f, 5
C-c M l, 5
C-c M m, 5
C-c M t, 5
C-c M v, 5
C-x C-0, 10

Variable Index

niori-appearance-font-family, 10
niori-appearance-manage, 10
niori-attachments-external-opener, 9
niori-attachments-import-method, 9
niori-data-directory, 9
niori-data-file, 9
niori-documents-directory, 9
niori-encryption-recipients, 10
niori-fhir-server-url, 10
niori-letter-flyspell, 10
niori-standalone-cua, 10
niori-store-keep-revisions, 9
niori-store-tighten-data-directory, 9
niori-vaccinations-due-days, 9
niori-visits-upcoming-days, 9

Function Index

niori, 5
niori-dashboard, 6
niori-export-org, 8
niori-fhir-export, 8
niori-record-ediff, 8
niori-store-backup, 8
niori-store-switch-data-directory, 13

Concept Index

appearance, 10
attachments, 7
attachments, safety, 13
audit trail, 13

backups, 8

cancel a command, 17
completion, 18
concepts, 2
Control key, 17
customization, 9

dashboard, 6
data directory, 12
detail views, 7
documents, 7

EasyPG, 19
echo area, 18

emacs -Q, 19
Emacs vocabulary, 18
Emacs, introduction, 17
encryption, 13
environment variables, 10
exports, 8

FHIR, 8
file permissions, 12
first run, 4
font size, 10
forms, 6

GNU ELPA, 18
GPG, 19
gpg-agent, 19

help, built-in, 17
history, 8

init file, 18
init file, skipping, 19
installation, as a package, 3
installation, standalone, 3
integrations, 11

key notation, 17
key pair, 19
keymap, 18

lab tests, 7
letters, 7

medical records, 1
medications, 7
MELPA, 18
Meta key, 17
minibuffer, 18
modules, 1

network, 13

open a file, 17

packages, 18
passphrase, 19
permissions, 12
prefix argument, 17
privacy, 13

relocation, 13
requirements, 2
revisions, 13
rosters, 6
running commands by name, 17

schema migrations, 14
search, 18
standalone application, 3
survival commands, 17

text scale, 10
themes, 10

undo, 17
upgrading, 4
user options, 9

vaccinations, 7
validation, 6
variables, 9
visits, 7