# PublisherStudio documentation

## Version 2.2.2

Tracked source documentation snapshot. The owner-side release build replaces this file with the complete DocFX HTML-backed PDF and generated API reference.

*index.md*

# PublisherStudio documentation

Version 2.2.2

PublisherStudio is a local-first publishing workspace for page layout, stories, spreadsheets, pictures, audio, video, streaming, interactive content, and self-contained exports.

This site is the maintained product and architecture documentation. It uses the same documentation shell as LocalGPT, so navigation, search, themes, side rails, and generated API pages behave consistently. ■

## Choose a path

### ■ *Use PublisherStudio*

Create a publication, learn the workspace, and understand pages, objects, stories, ribbons, and export workflows.

Open the user guide (articles/getting-started.md)

### ■ *Build rich content*

Work with stories, spreadsheets, pictures, audio, video, animation, interaction, streaming, and recording.

Explore pictures and media (articles/pictures-and-media.md)

### ■■ *Build and maintain it*

Read the modular architecture, development requirements, installer behavior, documentation pipeline, and release checks.

Open engineering guidance (articles/developer-build.md)

### ■ *Look up details*

Browse publishing and export behavior, privacy boundaries, the optional LocalGPT connection, and the generated API reference.

Browse reference material (articles/documentation-system.md)

## Architecture at a glance

```
flowchart LR
    U[Human author] --> UI[Blazor + DevExpress workspace]
    UI --> APP[Application services]
    APP --> DOC[Publication model]
    APP --> MEDIA[Media and streaming services]
    APP --> EXPORT[Print, PDF, web and recording exports]
    APP --> WIRE[Optional LocalGPT 1-Wire boundary]
    DOC --> STORE[(Local project data)]
    MEDIA --> DEVICE[Approved local devices]
    WIRE --> PEER[Approved LocalGPT peer]
```

The editable publication remains authoritative. Browser runtimes accelerate interaction and rendering, while C# services own project state, validation, persistence, export, and approved external connections.

## Complete documentation set

The conceptual pages are built together with compiler-generated XML documentation for public types and members. The same themed HTML site is shipped inside PublisherStudio and published to GitHub Pages.

<a class="btn btn-primary" href="PublisherStudio-2.2.2.pdf" download>■ Download the Kawaii handbook</a>

*articles/animation-and-interaction.md*

# Animation and interaction

Animations belong to the publication model, not to one export format.

Each object can own entrance, emphasis, motion, media, and exit steps. Pages can own transitions, and objects can respond to clicks or presentation events.

## Timeline rules

Animation order is page-wide. This gives the timeline one deterministic sequence across text, pictures, shapes, media, panels, and data visuals.

## Presentation actions

Interactions can navigate pages, open safe URLs, show or hide targets, replay animations, and control media. An object can be hidden at presentation start while remaining visible and editable in the authoring workspace.

## Export mapping

The HTML runtime maps semantic animation records to browser animations. Print and static images show the authored visual state. Video export renders the timeline. Unsupported behavior should be reported as a capability or loss marker rather than silently disappearing.

*articles/architecture.md*

# Architecture

PublisherStudio is a dependency-injection-oriented modular monolith. The design keeps the application easy to ship while giving each responsibility one clear home. ■

## Main roots

• Components own UI state and user interaction.

• Controllers own backend HTTP request entry points and results.

• Hubs own persistent connection entry points.

• HostedServices own application-lifetime scheduling.

• Services own reusable processing, persistence, devices, files, networks, FFmpeg, and operating-system work.

• BusinessObjects own authoritative documents, requests, state, and result contracts.

Reusable work flows toward Services and BusinessObjects. Services do not depend on Components, Controllers, Hubs, or HostedServices. Specialized use cases stay in a UseCases subnamespace below their owning service area rather than becoming a competing application root.

```
flowchart LR
  UI[Blazor Components] --> S[Reusable Services]
  C[Controllers: backend request entry] --> S
  HB[Hubs: persistent connection entry] --> S
  HS[HostedServices: lifetime work] --> S
  S --> BO[BusinessObjects]
```

## Service-owned behavior

Program.cs is the host composition root. Runtime state and reusable behavior belong to DI-owned services. PublisherStudio does not add application convenience statics to bypass composition.

P/Invoke and native exports belong behind injected lifetime services. Records, DTOs, constructors, and other BusinessObjects describe data; they do not acquire files, devices, networks, loggers, or host state.

## Canonical content

The native publication model is authoritative. External formats are adapters:

```
external input → parse → temporary canonical model → validate → commit
canonical model → capability analysis → map → external output
```

Every semantic contract has one authoritative declaration. Service-local copies of BusinessObjects are not allowed.

## Browser boundary

C# owns canonical state. JavaScript owns high-frequency browser interaction and commits bounded results. Listeners, observers, object URLs, pointer capture, and media streams are released on rebind or disposal.

Editor interaction overlays live in a local positioned stacking context. They never change publication layer order, and a local editor problem is not solved with an application-wide z-index.

## Safe input boundaries

XML importers prohibit DTD processing and external resolvers. Archives validate paths before extraction. New packages, native binaries, or helper processes require an explicit architectural reason and release review.

*articles/developer-build.md*

# Developer build

## Requirements

• .NET 10 SDK

• Windows PowerShell 5.1 or newer for the maintained scripts

• Node.js for preparing licensed DevExpress browser assets and documentation PDF generation

• A licensed DevExpress build identity for preparing runtime assets

## First source build

```
.\Prepare-DevExpressAssets.cmd
.\Build-LocalDevelopment.ps1 -Configuration Debug
```

The preparation command restores the pinned browser packages and generates the public runtime license. The private DevExpress license remains on the build machine.

## Release build

```
.\Build-Release.ps1 -Runtime win-x64
```

All supported runtimes can be built sequentially with:

```
.\Build-AllRuntimes.ps1
```

The release lane restores the authoritative LocalGPT wire protocol package, publishes the application and standalone setup, validates configuration and documentation, then creates manifest-backed archives.

## Quality gates

The maintained build checks architecture, diagnostics, localization, InteractiveServer render modes, async continuations, component boundaries, text ownership, iterator policy, system-variable initialization, JavaScript diagnostics, publish profiles, installer workflow, XML documentation coverage, and documentation output.

## Release record

Each release keeps one concise changelog and one validation report. Old work diaries and personal progress notes are not part of the public documentation. Completed, partial, and deferred work is summarized only when it helps users or maintainers understand the shipped state.

*articles/documentation-system.md*

# Documentation system

PublisherStudio documentation follows the same release shape as LocalGPT while using PublisherStudio names, routes, and source layout.

## Inputs

• maintained Markdown under docs/articles;

• public and protected C# XML comments from PublisherStudio.Web.xml;

• the built PublisherStudio.Web.dll for API metadata.

## Outputs

• searchable DocFX HTML under wwwroot/help-docs;

• PublisherStudio-<version>.pdf;

• generated API pages;

• documentation-status.json;

• a searchable XML-comment catalog exposed by the running app.

## Kawaii design guide

The website and PDF use the same soft pink, violet, and warm neutral palette. Decorative motion respects reduced-motion preferences. Light, dark, and system themes are selectable from a dedicated control that remains above the navigation layer.

The decoration is a smile, not a fog machine: headings stay clear, code remains readable, and technical warnings keep their seriousness.

## GitHub Pages

GitHub Pages uses the same pinned-snapshot workflow as LocalGPT. The tracked repository-root .github/pages/publisherstudio-kawaii-docs.zip is validated for path safety, the persistent theme selector, the cat-paw favicon, relative project-page links, and API pages before deployment.

After a successful application build, run Update-GitHubPagesSnapshot.cmd to refresh the publication payload from the exact generated wwwroot/help-docs tree. The command repairs DocFX namespace landing pages, validates all remaining local links, and updates both the authoritative .github/pages/publisherstudio-kawaii-docs.zip workflow snapshot and the repository-root /docs no-Jekyll mirror. The mirror keeps branch-based Pages configuration operational while the workflow publishes the same validated content when Pages is configured for GitHub Actions.

*articles/editor-workspace.md*

# Editor workspace

The editor keeps page structure, object order, and reusable tools in one view.

## Main areas

• Title bar — publication name and save state.

• Ribbon — create, arrange, animate, stream, export, and open help.

• Pages pane — page selection and ordering.

• Mainframe — the active publication page.

• Inspector — properties for the selected page or object.

• Timeline — animation and media timing when enabled.

## Select and arrange objects

Click an object to select it. Use Ctrl or Shift for multi-selection. The Arrange commands move objects through the page layer stack, align them, or group them.

Snapping can use the grid, guides, page edges, and other objects. Rulers and guides are workspace aids; they are not exported as publication content.

## Interaction ownership

A gesture has one owner. The Mainframe owns ordinary page gestures. Picture, media, panel, and spreadsheet studios own their local editing gestures while open. Temporary handles, masks, and playheads never become publication layers.

## Safe cancellation

Closing or cancelling a studio leaves canonical page placement and layer order untouched unless you explicitly applied a result. This separation is one of the reasons the editor can support many object types without each tool inventing its own page model.

*articles/getting-started.md*

# Getting started

PublisherStudio opens directly into the publication workspace. A new document contains a page, a canvas, a ribbon, a page list, and a properties panel.

## Your first little publication

1. Give the publication a clear name in the title bar.
2. Choose Insert and add a text box, picture, spreadsheet, panel, or media object.
3. Move and resize the object on the page.
4. Use the right-hand inspector to adjust the selected object.
5. Save the native project before exporting.
6. Choose File and create a PDF, image, website, or recorded presentation.

## The native project is the source of truth

PublisherStudio stores editable publication data in its own project format. Exported images, websites, PDFs, and videos are delivery formats. Keep the native project when you want to edit the work later.

## Choose the application language

Use the language selector in the upper-right corner of PublisherStudio. The choice reloads the current local page, stores a request-culture cookie, and applies one complete JSON catalog to the application shell.

Application language and publication language are separate: the global selector changes PublisherStudio, while the Inspector language setting controls exported document and website metadata.

## Local by default

The application listens on the local loopback interface. It does not need LocalGPT to create or publish documents. The optional LocalGPT page remains quiet until you choose to connect it.

> [!NOTE]

> The first source build needs the prepared DevExpress browser assets. End-user release packages already contain the runtime assets they need.

*articles/installer-and-updates.md*

# Installer and updates

PublisherStudio uses the same simple deployment shape as LocalGPT. The setup is a one-click operation and keeps every runtime beneath one product-owned AppData folder.

## Installed layout

The installation root is:

```
%LOCALAPPDATA%\PublisherStudio
```

Application and setup releases keep their runtime wrappers inside that root:

```
PublisherStudio\
  winx64\
    PublisherStudio.Web.exe
  setupwinx64\
    PublisherStudio.Setup.exe
    Install.cmd
    Update.cmd
    Start.cmd
```

The wrapper names vary by operating system and architecture, but the product root does not move to Programs and does not reuse the former BlazorPublisher application folder.

## One-click behavior

Double-clicking PublisherStudio.Setup.exe performs the normal install/update path:

1. download the matching application release;
2. extract it into %LOCALAPPDATA%\PublisherStudio;
3. download and refresh the matching setup release;
4. ensure FFmpeg is available;
5. create the required Desktop and Start Menu shortcuts;
6. start PublisherStudio on port 58071.

No command-line argument is required.

The setup queries the newest published GitHub release and requires exact runtime assets. On Windows x64 those assets are winx64.zip and setupwinx64.zip; a missing pair is an error rather than permission to install another architecture.

## Updates

Normal updates extract reviewed release files over the existing product root. The root is not deleted unless --force-delete is explicitly supplied. Files that are not part of the incoming release remain untouched.

When setup is started from its installed setup folder, it continues from a temporary copy before extraction. This lets the installed setup executable and launchers be replaced during the same one-click update.

## Required shortcuts

Both Desktop and Start Menu receive these entries:

- PublisherStudio Install;
- PublisherStudio Update;
- PublisherStudio Start;
- PublisherStudio Folder.

The Install, Update, and Start entries call the checked-in command files from the matching setup runtime folder. The Folder entry opens %LOCALAPPDATA%\PublisherStudio directly.

## Compatibility

Former --*-blazorpublisher command-line names remain accepted so an older launcher can invoke the repaired setup. New launchers use the PublisherStudio names.

*articles/localgpt-and-onewire.md*

# LocalGPT and 1-Wire

PublisherStudio works without LocalGPT. The 1-Wire connection is an optional local collaboration path.

## Discovery and connection

PublisherStudio listens for compact LocalGPT discovery broadcasts on UDP port 51141. A selected peer can then establish the larger approved transport over TCP port 51140.

Discovery does not grant control. The user must approve the link, and each protected eye, hand, screen, spreadsheet, media, or publishing capability remains subject to its permission policy.

## Default ports

- PublisherStudio web host: 58071
- LocalGPT 1-Wire TCP: 51140
- LocalGPT discovery UDP: 51141

These contracts stay separate so both applications can run side by side.

## Protocol ownership

The shared wire protocol is versioned independently and consumed from the authoritative LocalGPT.WireProtocolVersion package. PublisherStudio does not carry a second copy of the protocol project.

## Quiet standalone behavior

When LocalGPT is absent, PublisherStudio continues normally. The Organic Plugins page reports that no peer is connected instead of treating the optional capability as an application failure.

## Runtime identity and small devices

PublisherStudio identity and secrets are generated at runtime; they are not shipped as a shared default credential. Small peers such as an ESP32 can participate only through an approved capability and transport profile. Discovery alone never grants an ESP32—or any other peer—permission to capture, publish, or control content.

*articles/pictures-and-media.md*

# Pictures and media

Picture Studio and Media Studio keep editing details inside the selected publication object.

## Picture Studio

Picture documents can contain raster, text, shape, fill, and procedural layers. Crop, masks, transparency, filters, tint, blend modes, drawing, and polygon selections remain editable until you apply the result.

PublisherStudio stores both a rendered image and the editable picture document. The rendered image keeps normal publication export simple; the picture document lets you return for non-destructive edits.

## Video and audio

Media objects use ordered segments. You can trim, split, merge, copy, replace, and arrange segments without changing the object's page position or layer order.

Video regions are stored as normalized source coordinates, so they remain aligned across different source sizes. Audio uses time only and does not pretend to have a two-dimensional region.

## Recording preview

The live preview keeps the active browser media stream attached to the preview element. If Blazor replaces the video element, a bounded watchdog reattaches the stream. Saved recording data stays separate from the preview surface.

## Local editing overlays

Internal media sections and selection overlays never become page siblings or picture layers. They are transient editor projections inside a local positioned stacking context. The publication object remains the canonical content, while browser-side pointer feedback stays lightweight and disposable.

## Open picture and publication interchange

PublisherStudio keeps editable source information when it can and reports losses when a target format cannot represent a feature. Supported adapter families include SVG / SVGZ, OpenRaster, OpenDocument Drawing, and OpenDocument Presentation. The native PublisherStudio project remains the complete editable source.

## Open video projects

Project format is not a media codec or container. PublisherStudio can inspect and import project structures from OpenTimelineIO, MLT XML, Kdenlive, Shotcut, XGES, OpenShot, and CMX 3600 EDL. OBS Scene Collection data may be used as source context where supported.

The first editable projection is the active video-track projection, not full multitrack compositing. Missing media is reported for relinking instead of silently discarded, and the original project remains the safest archival copy. ■■

Audio interchange can preserve production metadata through Broadcast WAV where the selected import or export path supports it.

*articles/privacy-and-security.md*

# Privacy and security

PublisherStudio is local-first, but local software still needs careful boundaries.

## Data ownership

Publication projects, recordings, workbooks, pictures, settings, and protected connection state remain under the signed-in user's account. Exporters do not upload content by default.

## Untrusted input

Imported archives, SVG/XML, websites, media, JSON, spreadsheet files, LocalGPT envelopes, and browser messages are treated as untrusted. Importers validate paths, sizes, required files, and dangerous content before committing changes.

## Credentials

OAuth sessions, stream keys, LAN secrets, and machine-specific capture settings are stored outside publications and templates. They are not copied into normal interchange exports.

## Loopback host

The default web endpoint binds to 127.0.0.1. Opening PublisherStudio to another interface is a deliberate deployment decision and should be paired with appropriate authentication, firewall, and transport controls.

*articles/publishing-and-export.md*

# Publishing and export

PublisherStudio can deliver one project in several forms.

## Print and PDF

Print uses the publication page model and browser print surface. Choose Print / Save as PDF when the target is a fixed document.

## Images and SVG

PNG and JPEG are raster outputs. SVG keeps vector-friendly content where possible and freezes or marks media that cannot remain interactive.

## Websites

• Interactive presentation HTML keeps page navigation and interaction.

• Single-file website embeds the delivery payload into one HTML file.

• Structured website ZIP creates a normal file tree with content-addressed assets and the PublisherStudio runtime.

The structured exporter validates archive paths, preserves original media when optimization fails, and does not upload project data.

## Recorded presentation

Video export renders publication timing into a media file. Keep the native publication beside the export so the sequence can be revised later.

> [!TIP]

> Publish for the audience, but save for your future self. The native project is the editable keepsake. ■

## Structured website media policy

Structured websites avoid Base64 overhead by writing normal content-addressed assets. Preserve source media whenever possible: the exporter prefers to keep it when the browser can use it safely. Optional conversions may target PNG, WebP, AVIF, WebM, or lossless FFV1 workflows, depending on the source and available tooling.

Browser conversion is coordinated through Blazor and JavaScript interop. FFmpeg.wasm can be an optional browser-side helper, but export does not depend on it: when optimization is unavailable or fails, PublisherStudio keeps the compatible source instead of damaging the publication.

*articles/stories-and-spreadsheets.md*

# Stories and spreadsheets

PublisherStudio embeds rich documents and workbooks as normal publication objects.

## Rich stories

Story editing uses DevExpress RichEdit. A story can keep formatted text, page settings, fields, backgrounds, and document content inside the publication.

Apply saves the edited story back to the selected text frame. Cancel leaves the existing object unchanged.

## Spreadsheets

Spreadsheet Studio uses the DevExpress ASP.NET Core Spreadsheet in a same-origin editor surface. Workbooks can be created or imported, edited, downloaded, and applied back to a spreadsheet frame.

The page canvas and exports use a safe preview of the workbook. Executable workbook content is not injected into the publication DOM.

## Practical tips

- Keep source workbooks when macros or unsupported features matter.
- Use Fit when a whole sheet should remain visible.
- Use Clip when the frame is a deliberate viewport.
- Reopen a spreadsheet frame to continue editing the embedded workbook.

*articles/streaming-and-recording.md*

# Streaming and recording

Streaming Studio combines scenes, live inputs, chat, hotkeys, recording, and output control in one local session.

## Session flow

1. Open Streaming Studio.
2. Choose or create a profile.
3. Add page, camera, screen, window, audio, or browser sources.
4. Run a dry test.
5. Start recording or a configured output.
6. Stop the output before closing the session.

## Local capture

Browser capture is used where it fits. Windows process-loopback capture and FFmpeg services are selected only on supported platforms. Credentials and stream keys stay in protected local stores, not inside publications or interchange files.

## Preview and saved media

The preview is a live projection. The recording pipeline owns the saved bytes. A preview that needs reattachment should not corrupt a complete recording.

## When the preview turns black

First check whether the saved recording is still complete. If it is, reopen the preview or change the active source. The diagnostics bridge records browser-side failures, while expected circuit disconnects remain low-noise diagnostics.

*articles/troubleshooting.md*

# Troubleshooting

## The source builds but the browser UI is incomplete

Run Prepare-DevExpressAssets.cmd. A clean source package intentionally omits generated licensed DevExpress runtime assets.

## The application reports a port override warning

PublisherStudio uses the configured Kestrel endpoint as the authority. The maintained default is 127.0.0.1:58071. Remove conflicting ASPNETCORE_URLS or launch-profile addresses when you want a quiet startup log.

## LocalGPT is not discovered

Confirm that both applications are running, UDP port 51141 is available, and the firewall allows local discovery. PublisherStudio remains usable while discovery is unavailable.

## A browser circuit disconnects

OperationCanceledException and JSDisconnectedException can occur while a tab reloads or closes. They should be logged as expected disconnect diagnostics, not presented as data-loss errors. Reopen the page and check the application log for a preceding operational failure.

## Recording preview becomes black

Check the saved recording first. If the file is complete, reopen or reselect the preview source. The preview-rebind watchdog is designed to repair a replaced video element without changing the saved recording pipeline.

## Documentation is missing

A normal release build generates wwwroot/help-docs. The in-app Documentation page shows HTML, PDF, and XML-comment availability. Developer builds can diagnose the documentation lane separately with the build properties described in Documentation system (documentation-system.md).