

Artlogic CMS Settings Audit/Resource

Summary & Project Goals

This initiative aims to establish a centralized knowledge base for Artlogic CMS settings by bridging technical codebase metadata with Client Liaison insights. Developers are responsible for engineering parsing and Markdown generation scripts to automatically extract and structure settings data. Following a collaborative review phase with Client Liaisons to perfect the documentation, the dataset will be fully compiled and used by the Client Liaison team to add context to settings and query against using AI agents. Future development opportunities include integrating live Jira bug tracking and building a CMS help overlay for in-system dev access to the resource document.

This resource will serve three core functions:

- **Internal Team Resource:** A directory for Client Liaisons to understand settings, troubleshoot behaviors, and log known edge cases.
- **Settings Audit:** A review of current behaviors, implementation status, and identification of redundant or legacy settings.
- **AI Knowledge Base:** A clean, machine-readable dataset formatted for Retrieval-Augmented Generation (RAG) and LLM prompting to aid support and technical query responses.

Standardized Setting Block Schema

To prevent "data drift" across hundreds of CMS settings, every setting entry will be generated using a dual-purpose template that isolates developer-owned technical data from support-owned functional notes.

Below is a draft outline of this schema. This draft was generated using browser developer tools and Gemini AI. This will need to be edited and refined by developers with access to the code base.

None

```
### {Setting UI Label}
#### Technical Metadata
- **Setting Key:** `{setting_key_id}`
- **CMS Path:** `CMS > Settings > Settings > {Section_Name}`
- **Control Type:** Checkbox | Dropdown | Text Input | Radio
- **Default Value:** `{default_value}`
- **Prerequisites:**
  * `{Parent_Module_Name}` (`{parent_setting_key}` =
```

```

`{Required_State}`)
- **Database Field:** `{table_name}.{column_name}` (`{data_type}`)
- **Visibility Scope:** Public Front-End | Admin Only | Dev Only
- **Access Level Required:** Administrator | Super Admin
- **Tags:** `#active` | `#deprecated` | `#themeSpecific` |
`#notRecommended` | `#highBugRisk`
- **Code Last Modified:** `{YYYY-MM-DD}` (Commit `{commit_hash}`)

#### Theme Overrides & Variances
- **Default Value Overrides:**
  * `{Theme_Name_1}`: Defaults to `{override_default_value_1}`
  * `{Theme_Name_2}`: Defaults to `{override_default_value_2}`
- **Option Restrictions & Badges:**
  * `{option_label}`: `[Theme_Name_1 Only]` | `[Not Available
on Theme_Name_2]`
- **Behavioral Differences:**
  * `{Theme_Name_1}`: `{Description of theme-specific rendering or
fallback logic}`

#### Available Options ({count})
- `{option_1_label}` (`{option_1_value}`)
- `{option_2_label}` (`{option_2_value}`)

#### Cascading Dependencies & Component Impacts
- **Child Settings:**
  * `{Child_Setting_Label}` (`{child_setting_key}`)
- **Codebase References:**
  * Template View: `{path_to_template}`
  * API Controller: `{path_to_controller}`

**Functional Description**
{Human-readable explanation of what this setting controls on the
live site or back-end.}

> 💬 **Support Notes & Known Issues**
Type support observations, client bug reports, or edge cases here.
Comment format must list commenter's name, date (year, month, day),
and comment.
>
>
> Example:
> Stephen Mercer | 2026-08-18: Lorem ipsum doler sit amet...

```

Implementation Roadmap

Phase	Phase Name	Primary Actions	Key Deliverable / Milestone
Phase 1	Code Parsing & Extraction	Parse backend form definitions, AST templates, and DB schemas	Raw extracted JSON containing technical metadata & conditional logic
Phase 2	Baseline Document Export	Run script to generate standardized Markdown files	Technical baseline <code>.md</code> documents with anchor-delimited zones
Phase 3	CL Trial & Review	Client Liaisons access limited trial resource, add descriptions, support notes, & edge cases	Find hindrances, errors, and roadblocks to refine document generation and export
Phase 4	Refinement and re-export	Developers refine based on Client Liaison's findings and re-export the documents	Resolve issues and generate final documents for implementation.
Phase 5	Knowledge Base Deployment	Import Markdown files into central repository (Notion, GitHub, or Coda)	Searchable, team-wide settings directory with inline commenting
Phase 6	Human Enrichment & Review	Client Liaisons add functional descriptions, known edge cases, and	Complete audit files

		troubleshooting notes	
Future Development			
Phase 7	Jira & Issue Syncing	Link setting keys to Jira bug tracking fields and query feeds	Live issue tracking mapped directly to specific setting blocks
Phase 8	CMS Overlay & AI RAG Integration	Connect enriched Markdown to in-CMS support drawer & AI model prompt feeds	In-app contextual help drawer & automated AI support agent

Phase 1: Code Parsing & Extraction

- **Technical Implementation:** Developers use a script (ie. Node.js or Python) to extract metadata—including keys, types, and defaults—from backend files. The team then performs static code analysis (ie. grep or AST parsing) to trace variable dependencies, finally mapping the front-end impact by logging template and component references to link configuration to the UI.
- **Output:** A unified JSON dataset mapping every CMS setting, its technical metadata, and component relationships.

Phase 2: Baseline Document Export

- **Technical Implementation:** The extracted JSON payload is passed through a Markdown generator script that builds standardized `.md` files according to the established dual-layer schema. The script automatically inserts anchor-delimited markers (such as blockquote delimiters >  **Support Notes**) to establish protected zones for human contributions.
- **Output:** A complete repository of machine-generated baseline Markdown files with full technical metadata, CMS navigation paths (CMS > Settings > [Section]), codebase references, and theme flags.

Phase 3: CL Trial & Review

- **Technical Implementation:** A target subset of core module documentation (e.g., *Artworks*, *Artists*, *Exhibitions*) is deployed to a select group of Client Liaisons for trial auditing. Liaisons test writing **Functional Descriptions** and logging real-world client edge cases (e.g., missing dimensions breaking View on wall scaling).
- **Output:** Practical feedback identifying workflow roadblocks, missing metadata fields (like front-end scope or permission levels), and script formatting errors.

Phase 4: Refinement and Re-export

- **Technical Implementation:** Engineers update the extraction scripts based on trial findings—adding missing attributes like navigation paths and refine parsing logic for theme variance badges (e.g., **[Aperture Only]**). The re-extraction script is executed to verify that machine updates overwrite technical blocks without destroying human-written support notes in the trial files.
- **Output:** A fully validated extraction pipeline and refined master Markdown export.

Phase 5: Knowledge Base Deployment

- **Technical Implementation:** Baseline Markdown files are programmatically imported into the central team repository (ie. Notion, GitHub, or Coda). Navigation hierarchies are configured to mirror the Artlogic CMS sidebar tabs, and permission structures are set up to allow inline commenting across Support, Product, and Engineering.
- **Output:** A searchable, version-controlled internal knowledge base accessible across all departments.

Phase 6: Human Enrichment & Review

- **Technical Implementation:** Client Support and Liaisons systematically populate the **Functional Description** and **Support Notes & Known Issues** sections across all 22+ module files. Staff enforce standardized tag taxonomies (**#deprecated**, **#highBugRisk**, **#themeSpecific**) and strict comment attribution formatting (**[Name] | [YYYY-MM-DD]: [Details]**).
- **Output:** Production-ready documentation combining code-level precision with practical support guidance.

Phase 7: Jira & Issue Syncing (*Future Development*)

- **Technical Implementation:** A custom **Setting Key** field is deployed across internal Jira project boards. A bi-directional API script or webhook queries active and resolved Jira tickets tagged with setting identifiers (e.g., **roomview_wall**) and injects a live feed of active bug tickets directly under the corresponding setting block in the knowledge base.

- **Output:** Real-time visibility into open software bugs and regression reports directly within the setting documentation.

Phase 8: CMS Overlay & AI RAG Integration (*Future Development*)

- **Technical Implementation:** Enriched documentation data is exposed via a secure API endpoint. For human staff, a browser extension or native CMS script renders an in-app slide-out help drawer when hovering over setting labels inside the Artlogic admin panel. Concurrently, the cleaned Markdown repository is ingested into an AI vector database (RAG pipeline) to power automated support tools and contextual setup assistants.
 - **Output:** In-context CMS guidance for internal teams and a machine-readable knowledge base for AI-assisted support workflows.
-

Framework, Questions, and Future Development

Q1: Theme Management Strategy

Problem: Differentiation between themes becomes a large hurdle in support. Possible solution, duplicating documents per theme, creates severe maintenance overhead and leads to "data drift."

Solution: Maintain a Single Master Document with Theme Delta Overrides.

- **Unified Baseline:** Maintain one core document for all settings shared across themes.
- **Theme Delta Sub-Blocks:** When a setting behaves differently or defaults to a distinct value per theme, note it within a dedicated Theme Overrides block rather than duplicating the entire setting entry.
- **Inline Theme Badges:** Tag theme-exclusive settings or options directly in the list (e.g., [Aperture Only] or [Aspect Only]).

Q2: Standardized Tagging Taxonomy

Problem: Unstructured support comments make filtering and bulk auditing impossible.

Solution: Standardize Metadata Tags & Hashtags. Enforce a fixed set of tags inside the Technical Metadata block:

- **#deprecated** – Setting is legacy and scheduled for removal.
- **#notRecommended** – Setting exists for edge cases but often causes client layout issues.
- **#themeSpecific** – Behavior or availability depends on the active template theme.
- **#highBugRisk** – Frequently involved in client support escalations or regression bugs.
- **#devOnly** – Hidden or restricted toggle intended only for internal developers.

Q3: Division of Responsibility

Problem: Overlapping maintenance duties lead to unmaintained data and broken code links.

Solution: Establish a Domain Ownership Matrix.

Section / Component	Responsible Party	Maintenance Action
Technical Metadata	Developers	Automated script extraction on release
Prerequisites & Dependencies	Developers	Code-level AST parsing & validation
Codebase References	Developers	Automated path mapping
Functional Descriptions	Client Liaisons	Initial draft & client-facing tone refinement
Customer Support Notes	Client Liaisons	Ongoing logging of edge cases & client feedback
Tag Allocation	Joint Review (Support + Dev)	Quarterly audit of #deprecated & #notRecommended flags

Q4: Terminology Alignment

Problem: Developers use internal code keys

(`setting_field_artworks_detail_roomview_enabled`), while Support uses UI labels ("Enable View on a Wall").

Solution: Structure Entries with a Dual-Layer Hierarchy.

1. Primary Header (UI Layer): Use the user-facing CMS string as the main section header (### Enable View on a Wall (Roomview)).
2. Metadata Sub-Bullet (Technical Layer): Explicitly list the system key immediately beneath it (Setting Key: `setting_field_artworks_detail_roomview_enabled`).
3. AI Search Optimization: This dual mapping ensures non-technical staff can search by visual label, developers can search by database key, and AI models can bridge the gap between the two.

Q5: Automated Update & Re-Extraction Workflow

Problem: Document requires continual updating. Re-running developer extraction scripts risks overwriting human-written support notes and descriptions.

Solution: Implement Anchor-Delimited Parsing.

- Machine vs. Human Zones: The extraction script treats the Markdown file as two distinct zones divided by structural headers.
- Protected Blocks: The script is programmed to wipe and regenerate only the Technical Metadata, Prerequisites, Available Options, and Cascading Dependencies sections upon code updates.
- Preserved Blocks: Text residing beneath Functional Description and inside the >  Customer Support Notes blockquote is preserved line-for-line during automated re-extractions.

Q6: In-CMS Staff Integration

Problem: Navigating external documentation during live support calls creates friction.

Solution: Build an In-CMS Contextual Support Overlay.

- Database Storage: Store Functional Description and Customer Support Notes in a database table indexed by setting_key.
- Support Slide-Out Drawer: Inject a hidden [?] icon or slide-out drawer into the Artlogic CMS settings UI (visible only to authenticated internal staff). Clicking the icon pulls live support notes, prerequisites, and open Jira bugs directly into the CMS page.

Q7: Client-Side Help Resources

Problem: Public client documentation requires a different tone and level of detail than internal technical audits.

Solution: Deploy a Single Source of Truth Export Pipeline.

- Automated Sanitization: At export time, a build script strips out developer metadata (Database Field, Codebase References, Commit Hashes) and internal Support Notes.
- Public Generation: The remaining Setting Label, Available Options, and cleaned Functional Description text are pushed directly into public Help Center articles or in-app client tooltips (? icons).

Q8: Jira Ticket Integration

Problem: Issue tracking and documentation exist in siloes, causing duplicate bug reporting.

Solution: Map Jira Tickets via the Universal Setting Key.

- Custom Jira Field: Add a mandatory or optional Setting Key text field to internal Jira bug report templates.
- Dynamic Issue Feeds: During documentation generation or inside the CMS slide-out

drawer, run a lightweight API call to Jira to list open or recently resolved tickets directly under that setting's entry:

None

```
* **Open Jira Issues:** [PS-1042] Roomview canvas fails to load on Safari 18 (`roomview_wall`)
```

Markdown examples

Below are examples of four settings and their draft markdown schema.

Note: This is draft and not based on core code, rather best guesses from available resources. I did not generate sufficient data to complete the theme overrides block. This block has been included with the standardized schema values.

None

```
### Enabled Modules
```

```
#### Technical Metadata
```

```
- **Setting Key:** `multi_field_sections_enabled_{module_name}`
- **CMS Path:** `CMS > Settings > Settings > General`
- **Control Type:** Checkbox Collection / Toggle Group
- **Default Value:** Core system modules enabled by default
- **Prerequisites:** None (Master System Setting)
- **Database Field:**
`tbl_preferences.multi_field_sections_enabled_` (`tinyint(1)`)
- **Visibility Scope:** CMS Navigation Sidebar & Public Front-End
Navigation
- **Access Level Required:** Administrator
- **System Dependencies:** Core System Routing Architecture
- **Code Last Modified:** `2025-11-12` (Commit `#a8f9b2c`)
```

```
#### Theme Overrides & Variances
```

```
- **Default Value Overrides:**
  * `{Theme_Name_1}`: Defaults to `{override_default_value_1}`
  * `{Theme_Name_2}`: Defaults to `{override_default_value_2}`
- **Option Restrictions & Badges:**
  * `{option_label}`: `[Theme_Name_1] Only` | `[Not Available
on {Theme_Name_2}]`
- **Behavioral Differences:**
```

```
* `{Theme_Name_1}`: `{Description of theme-specific rendering or fallback logic}`
```

Available Options (22)

```
- `"Art fairs"` (`multi_field_sections_enabled_art-fairs`)
- `"Artist categories"`
(`multi_field_sections_enabled_artist_categories`)
- `"Artists"` (`multi_field_sections_enabled_artists`)
- `"Artwork categories"`
(`multi_field_sections_enabled_artwork_categories`)
- `"Artworks"` (`multi_field_sections_enabled_artworks`)
- `"Blog entries"` (`multi_field_sections_enabled_blog_entries`)
- `"Clients"` (`multi_field_sections_enabled_clients`)
- `"Events"` (`multi_field_sections_enabled_events`)
- `"Exhibitions"` (`multi_field_sections_enabled_exhibitions`)
- `"Feature panels"`
(`multi_field_sections_enabled_feature_panels`)
- `"Fonts"` (`multi_field_sections_enabled_fonts`)
- `"Home"` (`multi_field_sections_enabled_home`)
- `"News"` (`multi_field_sections_enabled_news`)
- `"Press"` (`multi_field_sections_enabled_press`)
- `"Products"` (`multi_field_sections_enabled_products`)
- `"Publications"` (`multi_field_sections_enabled_publications`)
- `"Store"` (`multi_field_sections_enabled_store`)
- `"Team"` (`multi_field_sections_enabled_team`)
- `"Testimonials"` (`multi_field_sections_enabled_testimonials`)
- `"Translations"` (`multi_field_sections_enabled_translations`)
- `"Video entries"` (`multi_field_sections_enabled_video_entries`)
- `"Viewing rooms"` (`multi_field_sections_enabled_viewing_rooms`)
```

Cascading Dependencies & Component Impacts

```
- **Child Settings:** All section-specific configuration panels
(e.g., Artists Settings, Exhibitions Settings)
- **Codebase References:**
  - Admin Sidebar: `views/admin/sidebar.tpl`
  - Header Navigation: `views/header/navigation.tpl`
```

****Functional Description****

A master collection of toggles that activates or deactivates specific website features and public sections. Disabling a module hides its corresponding navigation links on the live website and removes its record management interface from the CMS menu.

```
> 🗨️ **Support Notes & Known Issues**
Type support observations, client bug reports, or edge cases here.
Comment format must list commenter's name, date (year, month, day),
and comment.
>
>
> Example:
> Stephen Mercer | 2026-08-18: Lorem ipsum doler sit amet...
```

Caption Fields

Technical Metadata

```
- **Setting Keys:**
`multi_field_artwork_caption_fields_list_{field_name}`
- **CMS Path:** `CMS > Settings > Settings > Artworks`
- **Control Type:** Drag-and-Drop Checkbox List / Ordering
Organizer
- **Default Value:** Standard metadata fields enabled (`Artist`,
`Title and year`, `Medium`, `Dimensions`, `Edition details`)
- **Prerequisites:**
  - Artworks Module (`multi_field_sections_enabled_artworks` =
`Checked`)
- **Database Field:**
`tbl_preferences.multi_field_artwork_caption_fields_list_*`
(`tinyint(1)`)
- **Visibility Scope:** Public Front-End (`/artworks/detail`)
- **Access Level Required:** Administrator
- **System Dependencies:**
  - Module: Artworks (`multi_field_sections_enabled_artworks`)
- **Code Last Modified:** `2025-11-12` (Commit `#a8f9b2c`)
```

Theme Overrides & Variances

```
- **Default Value Overrides:**
  * `{Theme_Name_1}`: Defaults to `{override_default_value_1}`
  * `{Theme_Name_2}`: Defaults to `{override_default_value_2}`
- **Option Restrictions & Badges:**
  * `{option_label}`: `[Theme_Name_1] Only` | `[Not Available
on {Theme_Name_2}]`
- **Behavioral Differences:**
  * `{Theme_Name_1}`: `{Description of theme-specific rendering or
fallback logic}`
```

Available Options (8)

```
- `"Artist"` (`multi_field_artwork_caption_fields_list_artist`)
- `"Title and year"`
  (`multi_field_artwork_caption_fields_list_title_and_year`)
- `"Medium"` (`multi_field_artwork_caption_fields_list_medium`)
- `"Dimensions"`
  (`multi_field_artwork_caption_fields_list_dimensions`)
- `"Edition details"`
  (`multi_field_artwork_caption_fields_list_edition_details`)
- `"Series"` (`multi_field_artwork_caption_fields_list_series`)
- `"Signed and dated"`
  (`multi_field_artwork_caption_fields_list_signed_and_dated`)
- `"Collaborative artist"`
  (`multi_field_artwork_caption_fields_list_collaborative_artist`)
```

Cascading Dependencies & Component Impacts

```
- **Child Settings:**
  - Artwork Grid Caption Customization
    (`setting_field_artists_work_grid_full_details`)
  - Simplified Caption Style Formatting
    (`setting_field_artwork_grid_captions_simplified`)
  - Database Live Captions Sync
    (`setting_field_exhibitions_works_captions_from_artworks_database`)
- **Codebase References:**
  - Template View: `views/artworks/caption.tpl`
  - API Controller: `controllers/artworks/detail.php`
```

****Functional Description****

Controls which metadata attributes are displayed in text captions on artwork pages and organizes their display sequence.

> ****Support Notes & Known Issues****

Type support observations, client bug reports, or edge cases here. Comment format must list commenter's name, date (year, month, day), and comment.

>

>

> Example:

> Stephen Mercer | 2026-08-18: Lorem ipsum doler sit amet...

Artist List Layout

Technical Metadata

- ****Setting Key:**** `artists\_page\_defaults\_to`
- ****CMS Path:**** `CMS > Settings > Settings > Artists`
- ****Control Type:**** Dropdown / Select
- ****Default Value:**** `"Grid of images"`
- ****Prerequisites:****
 - Artists Module (`multi\_field\_sections\_enabled\_artists` = `Checked`)
- ****Database Field:**** `tbl\_preferences.artists\_page\_defaults\_to`
(`varchar(100)`)
- ****Visibility Scope:**** Public Front-End (`/artists`)
- ****Access Level Required:**** Administrator
- ****System Dependencies:****
 - Module: Artists (`multi\_field\_sections\_enabled\_artists`)
- ****Code Last Modified:**** `2025-11-12` (Commit `#a8f9b2c`)

Theme Overrides & Variances

- ****Default Value Overrides:****
 - * `{Theme\_Name\_1}`: Defaults to `{override\_default\_value\_1}`
 - * `{Theme\_Name\_2}`: Defaults to `{override\_default\_value\_2}`
- ****Option Restrictions & Badges:****
 - * `{option\_label}`: `[Theme\_Name\_1] Only` | `[Not Available on Theme\_Name\_2]`
- ****Behavioral Differences:****
 - * `{Theme\_Name\_1}`: `{Description of theme-specific rendering or fallback logic}`

Available Options (9)

- `"Theme default"`
- `"Grid of images"`
- `"List without images"`
- `"List with hover image (hover over list of names to reveal adjacent image)"`
- `"List with full-bleed background hover image"`
- `"List with hover image, randomly placed behind text"`
- `"A-Z list with hover image (hover over list of names to reveal adjacent image)"`
- `"Grid of images and names divided by artist categories"`
- `"List with hover image, divided by artist categories"`

Cascading Dependencies & Component Impacts

```
- **Child Settings:**
  - Artist List Preview Columns
  (`artists_list_preview_no_of_columns`)
  - Disable Autoplay for Preview Image Slideshow
  (`setting_field_disable_artist_list_slideshow_autoplay`)
  - Filter by Artist Categories on List
  (`setting_field_artists_enable_selected_categories`)
  - Split Artist List by Gallery and Exhibited Artists
  (`setting_field_artists_list_split_by_gallery_artists`)
- **Codebase References:**
  - Template View: `views/artists/index.tpl`
  - API Controller: `controllers/api/v2/artists.php`
```

****Functional Description****

Controls the primary visual layout and interactive presentation mode for the main Artists directory page.

> **Support Notes & Known Issues**

Type support observations, client bug reports, or edge cases here. Comment format must list commenter's name, date (year, month, day), and comment.

>

>

> Example:

> Stephen Mercer | 2026-08-18: Lorem ipsum doler sit amet...

Enable View on a Wall (Roomview)

Technical Metadata

```
- **Setting Key:** `setting_field_artworks_detail_roomview_enabled`
- **CMS Path:** `CMS > Settings > Settings > Artworks`
- **Control Type:** Checkbox / Toggle
- **Default Value:** `"Disabled"`
- **Prerequisites:**
  - Artworks Module (`multi_field_sections_enabled_artworks` = `Checked`)
- **Database Field:**
  `tbl_preferences.setting_field_artworks_detail_roomview_enabled`
  (`tinyint(1)`)
- **Visibility Scope:** Public Front-End (`/artworks/detail`)
- **Access Level Required:** Administrator
- **System Dependencies:**
```

```
- Module: Artworks (`multi_field_sections_enabled_artworks`)
- **Code Last Modified:** `2025-11-12` (Commit `#a8f9b2c`)
```

Theme Overrides & Variances

```
- **Default Value Overrides:**
  * `{Theme_Name_1}`: Defaults to `{override_default_value_1}`
  * `{Theme_Name_2}`: Defaults to `{override_default_value_2}`
- **Option Restrictions & Badges:**
  * `{option_label}`: `[Theme_Name_1 Only]` | `[Not Available on Theme_Name_2]`
- **Behavioral Differences:**
  * `{Theme_Name_1}`: `{Description of theme-specific rendering or fallback logic}`
```

Available Options (2)

```
- `"Enabled"`
- `"Disabled"`
```

Cascading Dependencies & Component Impacts

```
- **Child Settings:**
  - Roomview Mode (`artworks_detail_roomview_mode`)
  - Wall Style (`roomview_wall`)
  - Floor Style (`roomview_floor`)
  - Furniture Preset (`roomview_furniture`)
  - Enable View on a Wall Records
  (`setting_field_roomview_scene_records_enabled`)
- **Codebase References:**
  - Template View: `views/artworks/detail_roomview.tpl`
  - Component: `components/roomview/stage.js`
```

Functional Description

Enables the interactive "View on a Wall" staging feature on artwork detail pages, allowing website visitors to visualize artworks rendered to scale in a virtual room staging environment.

> **Support Notes & Known Issues**

Type support observations, client bug reports, or edge cases here. Comment format must list commenter's name, date (year, month, day), and comment.

>

>

> Example:

> Stephen Mercer | 2026-08-18: Lorem ipsum doler sit amet...

Wall Style (Roomview)

Technical Metadata

- ****Setting Key:**** `roomview\_wall`
- ****CMS Path:**** `CMS > Settings > Settings > Artworks`
- ****Control Type:**** Dropdown / Select
- ****Default Value:**** `"Standard"`
- ****Prerequisites:****
 - Artworks Module (`multi\_field\_sections\_enabled\_artworks` = `Checked`)
 - Enable View on a Wall
(`setting\_field\_artworks\_detail\_roomview\_enabled` = `Checked`)
 - Roomview Mode (`artworks\_detail\_roomview\_mode` = `"Standard"`)
- ****Database Field:**** `tbl\_preferences.roomview\_wall`
(`varchar(50)`)
- ****Visibility Scope:**** Public Front-End (`/artworks/detail`)
- ****Access Level Required:**** Administrator
- ****System Dependencies:****
 - Roomview Stage Component (`components/roomview/stage.js`)
- ****Code Last Modified:**** `2025-11-12` (Commit `#a8f9b2c`)

Theme Overrides & Variances

- ****Default Value Overrides:****
 - * `{Theme\_Name\_1}`: Defaults to `{override\_default\_value\_1}`
 - * `{Theme\_Name\_2}`: Defaults to `{override\_default\_value\_2}`
- ****Option Restrictions & Badges:****
 - * `{option\_label}`: `[Theme\_Name\_1] Only` | `[Not Available on {Theme\_Name\_2}]`
- ****Behavioral Differences:****
 - * `{Theme\_Name\_1}`: `{Description of theme-specific rendering or fallback logic}`

Available Options (4)

- `"Standard"`
- `"White paint"`
- `"Concrete"`
- `"Brick - White"`

Cascading Dependencies & Component Impacts

- **Child Settings:** None
- **Codebase References:**
 - Template View: `views/artworks/detail\_roomview.tpl`
 - Component: `components/roomview/wall\_renderer.js`

Functional Description

Defines the default wall surface texture applied behind artworks rendered within the interactive Roomview staging canvas.

> **Support Notes & Known Issues**

Type support observations, client bug reports, or edge cases here. Comment format must list commenter's name, date (year, month, day), and comment.

>

>

> Example:

> Stephen Mercer | 2026-08-18: Lorem ipsum doler sit amet...