Modern Mojo Inventory Cards .063

Purpose

This inventory-card document records the current Data plateau for Modern Mojo using the uploaded A-side exports, compare reports, manifest, and prior analysis doctrine. It is intentionally forward-looking and application-slicing aware.

Data plateau profile

The uploaded A-side export shows a shallow but active substrate, with the strongest live row counts concentrated in module and settings tables. Especially notable are `mp_ModuleSettings` (1117), `mp_ModuleDefinitionSettings` (398), `mp_ModuleDefinitions` (34), `mp_Modules` (26), `mp_PageModules` (10), `mp_Pages` (4), `mp_HtmlContent` (8), `mp_Place` (11), `mp_UserDevice` (2), and `mp_UserLocation` (100) ■filecite■turn49file0■

Card: mp_Pages

Table role: page identity and route-facing addressability.

Observed row count: 4

Key live fields: PageID, PageName, PageTitle, UseUrl, Url, ParentID, AuthorizedRoles, PublishMode, PageGuid ■filecite■turn42file0■

Connected chain: mp_Pages → mp_PageModules → mp_Modules → mp_ModuleDefinitions → mp_ModuleSettings / mp_HtmlContent

Re-tasking watch: fields like `UseUrl`, `Url`, `AuthorizedRoles`, and `PublishMode` suggest page identity is not only navigational but governance-bearing.

The small number of pages implies that each page is heavily loaded with intention rather than acting as generic CMS sprawl.

Card: mp_PageModules

Table role: page-to-module placement bridge.

Observed row count: 10

Key live fields: PageID, ModuleID, PaneName, ModuleOrder, GUID pairing ■filecite■turn42file0■

Connected chain: mp_PageModules is the hinge between page identity and module instance realization.

This table is one of the cleanest application-slicing pivots because it reveals how many separate operational surfaces are layered onto a small page set.

Card: mp_Modules

Table role: module instance carrier.

Observed row count: 26

Key live fields: ModuleID, ModuleDefID, ModuleTitle, Guid, FeatureGuid, IncludeInSearch, IsGlobal, PublishMode, DraftApprovalRoles ■filecite■turn41file0■ ■filecite■turn44file0■

Connected chain: mp_Modules provides the live instance record that lets definitions become actual page constructs.

Re-tasking watch: `FeatureGuid` and publication/governance fields widen old instance semantics into a richer B-side substrate.

This table feels like one of the most important projection tables: it turns structural possibility into live site composition.

Card: mp_ModuleDefinitions

Table role: render bridge from instance to control source.

Observed row count: 34

Key live fields: ModuleDefID, Guid, FeatureName, ControlSrc ■filecite■turn41file0■

Connected chain: mp_ModuleDefinitions tells which control actually renders the module.

Known control examples: HtmlModule, ContactForm, ImageGallery, HtmlFragmentInclude, CustomScript, IFrame-style surfaces ■filecite■turn41file0■

If a legacy form or construct exists in data but not on screen, this table is one of the first places to suspect a broken render bridge.

Card: mp_ModuleSettings

Table role: behavior/configuration substrate.

Observed row count: 1117

Key live fields: ModuleID, SettingName, SettingValue, ControlType, ControlSrc ■filecite■turn41file0■

Connected chain: mp_ModuleSettings appears to carry much of the compositional logic that a casual observer might wrongly expect to live in dedicated content tables.

This is one of the strongest signs that the old Portal is configured more like a platform of composed controls than a simple page-content CMS.

Card: mp_HtmlContent

Table role: HTML/body payload for content-bearing modules.

Observed row count: 8

Connected items: module-linked body content tied to specific ModuleID values ■filecite■turn41file0■

Re-tasking watch: where this table is sparse, actual behavior may be shifting into module settings, control-specific tables, or custom code paths.

The modest row count suggests that content exists, but the living application probably depends more on structure and configuration than on a large body of static editor content.

Card: mp_Place

Table role: custom domain place/address/location envelope.

Observed row count: 11

Known field model: RowID, UserGuid, SiteGuid, Name, Mode, ParentID, Description, Company, License, Addtl, Longitude, Latitude, Altitude, Address1, Address2, Continent, Country, Region, City, Zip, TimeZone ■filecite■turn53file1■

Connected items: Place is the likely anchor for address and place-style routes, especially residence/workplace analogs in the older slice analysis.

Place feels like a domain table that survived schema evolution better than many inherited CMS tables; its risk is not disappearance, but semantic under-projection in B-side routes.

Card: mp_UserDevice

Table role: custom domain device registry and operational vessel.

Observed row count: 2

Known field model: DeviceName, IMEI, SIMCard, HardwareVersion, IPPort, TrackName, TrackUser, TrackMail, GSM, GPS, Longitude, Latitude, DeviceStatus, TrackStatus, AlarmStatus, DeviceMode, AlarmMode, TrackMode, Emergency1-5, Messages, Tracks, Installed, LastCaptureUTC ■filecite■turn53file1■

Connected items: Factory, Assign, Install, and Tracker-family slices were already identified as likely UserDevice consumers.

UserDevice is still one of the clearest signs that the true application intent extends beyond stock mojoPortal into a fielded operations domain.

Card: mp_UserLocation

Table role: inherited or hybrid location/user substrate.

Observed row count: 100

The unexpectedly higher row count compared with Place and UserDevice suggests that inherited location machinery may still be carrying a surprising amount of live meaning in A, perhaps even where later custom tables seem conceptually more correct.

Card: mp_Tracking

Table role: downstream event/capture stream.

Observed row count: 0 in the manifest ■filecite■turn49file0■

Known field model: RowID, UserGuid, SiteGuid, DeviceGuid, Track, captureUTC ■filecite■turn53file2■

Tracking is structurally important even when empty in this snapshot, because it tells us where the system expects temporal device narrative to accumulate once the upstream registry world is alive.

Modern Mojo read

The data plateau suggests three overlapping strata:

1. page/module/content substrate

2. site/governance/config substrate

3. custom domain substrate

That overlap is likely why fields appear re-tasked. The system seems to re-use older substrate layers while projecting newer operational intent through them.