Cifru Documentation

Detailed guide for users and AI assistants. Read-only data workspace builder for iPhone, iPad, Apple Watch widgets, and supported CarPlay widget surfaces.

Version: July 11, 2026. Developer / seller: Dumitru Ionut-Madalin Dumitru. Contact: madalin@madalin.org.

Short Product Definition

Cifru lets a user build their own read-only mobile workspace from data they are authorized to access. The user installs the app, adds data sources, previews rows, maps fields, creates Home screens/buttons, adds row-level Details buttons, builds widgets, configures cache, and can switch into a simplified "Use as app" mode with their own company name and logo.

Cifru is universal. It can work with ERP data, ecommerce data, internal databases, public or private feeds, APIs, files, reports, and other structured data. It is not limited to a single ERP, shop platform, stock system, or business vertical.

Instructions For AI Assistants

If you are an AI assistant helping someone configure Cifru, use this document as the source of truth for available screens, buttons, fields, and limits.

- Always keep the setup read-only. Do not suggest INSERT, UPDATE, DELETE, POST, PUT, PATCH, mutation, write API calls, or credentials with write permissions.
- First ask what system the user wants to integrate, what access they have, and whether they have a database, API, feed file, FTP file, local file, or ecommerce API.
- Give instructions as exact navigation paths, for example: `Sources > Add sources > Databases > SQL Server`.
- Use only controls that exist in Cifru. If you need a missing control, say that Cifru does not currently expose it.
- Explain fields in normal language for non-programmers. Say "field", "column", "common field", and "data link" instead of assuming the user knows SQL joins.
- When telling the user to create a Details button, explain which parent field and child field must match, and ask them to use the matching-rate preview before saving.
- For sensitive systems, recommend a read-only user, VPN/private network access, TLS/HTTPS, and independent source-side backups/logging.
- If the integration target has public documentation, read it and map its available APIs, feeds, tables, or endpoints to one of the Cifru source types below.
- If the user asks you to generate an AI Configurator draft, return only valid declarative JSON that Cifru can validate. Do not return code, scripts, plugins, executable modules, or instructions that bypass Cifru controls.

Read-Only Safety Model

- Cifru is designed for read-only display, filtering, sorting, exporting, and dashboarding.
- Database lists are generated from controlled UI fields. Normal users do not type editable SQL inside Cifru.
- SQL-style requests should be used with source-side read-only database accounts.
- API and ecommerce connectors use read requests. GraphQL mutation and subscription operations are blocked.

- Cached rows, exports, imported files, and viewed business data are stored on the device and are not sent to a Cifru backend.
- External widgets use a small local snapshot. They do not connect directly to databases, APIs, feeds, files, or ecommerce systems.
- AI Configurator initially sends only safe schema metadata: internal source/list IDs, display names, field names, field types, allowed widgets, allowed filters, allowed actions, bundled Cifru connector behavior, and user-approved public vendor documentation context. If automatic repair needs to verify a relation and the user has left minimal mapping samples enabled, Cifru may send at most one strictly reduced identifier-only sample per relevant object (maximum 12 bounded fields). Credentials, tokens, URLs, names/contact data, invoices, descriptions, prices, totals, stock, quantities, and other business values are excluded locally. Exact schema discovered from the configured source remains authoritative.

App Navigation Map

Area	What the user sees	Purpose
Home	Brand header, workspace status, setup checklist, Home screens/buttons, widgets, empty state, Ask AI card.	The daily dashboard and the first onboarding surface.
Sources	Add sources button, plan usage, source rows with OK / Configure / Off status, source editors.	Where connections and credentials are created and tested. A source that was added but not successfully tested is visually marked Configure.
Build	Source readiness, compact Home-screen tree, nested Details and enrichments, Widgets, guided creation recipes, page Studio, and an optional Advanced library.	Where the user designs the workspace. Basic mode follows the visible result on the phone instead of exposing separate technical inventories. Advanced mode reveals the data library, relationship map, and common-field tools without removing any capability.
Settings	Subscription, language, theme, status, Siri and AI helper, cache, widgets snapshot, app mode, security, legal/help.	Device-level preferences, cache controls, security review, Siri shortcut hints, and subscription management.

Area	What the user sees	Purpose
Use as app mode	Configured Home screens/buttons and widgets, optional company logo/name header when the user set their own branding, exit path in settings.	Makes the configured workspace feel like the user's own simple app without showing a redundant generic Cifru header.

What Leaves The Device

Action	Destination	Payload	User control
Read a configured source	The database, API, feed, FTP server, shop, or object store chosen by the user.	Connection/authentication data and a fixed read-only query, GET/query request, or file path needed for that source. Cached rows and unrelated local data are not uploaded back.	The user creates, enables, refreshes, and can delete the source. Source-side permissions should be read-only.
Apple On-Device / Mock AI	No external AI provider.	Prompt and safe metadata remain local.	The user selects the provider or disables AI.
OpenAI, Claude, or Gemini	The provider selected and keyed by the user.	User prompt, optional system name, internal IDs, field names/types/capabilities, declarative JSON diagnostics, explicitly approved public documentation, and only during automatic repair an optional identifier-only mapping sample as described above. Never credentials, tokens, connection strings, private URLs, names/contact data, invoice contents, descriptions, prices, totals, stock, quantities, or full feeds.	Disabled unless configured. Cifru shows a per-request disclosure and sends only after the user confirms. Minimal mapping samples can be disabled in AI Configurator settings.
Export	The destination selected in Apple's share sheet.	The PDF/XLSX-style export chosen by the user.	No automatic developer upload.

Home Screen

First-install Demo button

On a genuinely new installation, Cifru adds one English-language **Cifru Demo (local)** Home example automatically. It demonstrates a customer-invoice header, a row-level **Details** section with invoice lines, and a **Product** sub-button on each line. Every row is sanitized and stored locally; the example never connects to a network source. The Demo source, three Demo lists, and two Demo relations are excluded from Free/Starter/Pro source, list, and Details-link usage, so the example cannot consume or unlock a plan entitlement. A visible DEMO badge distinguishes it from real screens. Deleting any Demo screen through its destructive action removes the complete bundled example and its local sample cache, without deleting or changing real sources. Once deleted, it is not recreated on ordinary launches.

The setup checklist intentionally ignores bundled Demo objects: it describes progress on the user's own source, fields, Home button, and Details relation. The existing **Start with an example** action remains available when the workspace is empty.

Header

Control / Text	Type	Purpose
Cifru logo + Cifru title	Brand header	Shows the app identity without using a plain, empty navigation title.
Subtitle: Your read-only data workspace	Static text	Reminds the user that the app is for read-only data work.
Plan pill: Free, Starter, or Pro	Status pill	Shows the active feature tier at a glance.

Workspace Status Card

Shows the latest app status message and four small counters: lists, rows, warnings, and stale items. Use this to tell the user whether Cifru has data, whether warnings exist, and whether refresh is needed.

Setup Checklist

The checklist is expanded while the workspace has no Home button. After the first Home button is created it collapses automatically to keep Home compact, and the user can expand or collapse it again at any time.

Checklist item	Meaning
Source tested	At least one source has answered successfully.
Data found	At least one list has preview or cached rows.
Fields chosen	At least one list has visible fields for rows or detail screens.
Home button	At least one list is exposed as a Home button.
Details added	At least one data link exists.

Empty State Buttons

Button	Type	What it does
Add first source	Button	Switches the user to Sources so they can add a real connection.
Start with an example	Button	Creates a local example workspace with demo lists, buttons, and widgets.
Open guide	Button	Opens the first-run guide.
Ask AI card - Copy prompt	Button + selectable text	Copies a localized AI prompt that points to the HTML documentation and asks an AI to guide a new integration. The card appears on the empty Home screen, in Start guide, in Settings, and can be opened through Siri/App Shortcuts.

Home Toolbar

Icon	Purpose
Question mark	Opens in-app Help.
Info	Opens Legal and privacy pages.
Refresh	Runs refresh for all configured lists and cached widgets.

Source Categories And Connectors

Path: `Sources > Add sources`. The Add sources button opens a menu grouped by category.

Category	Connectors	Best for
Databases	SQL Server, PostgreSQL, MySQL, MariaDB, Oracle	ERP databases, reporting databases, internal operational systems, read-only views.
Online shops	WooCommerce, Shopify, Magento / Adobe Commerce, PrestaShop, OpenCart custom read API	Orders, customers, products, stock, catalog data, store reports.
Data feeds	HTTP(S) feed, FTP file	CSV, TSV, JSON, XML files published by another system.
APIs	REST JSON, GraphQL, OData	Systems with documented read APIs.
Files	Local file	CSV, TSV, JSON, XML, or XLSX imported from the Files app for local viewing.
Object storage	S3-compatible storage	Structured files stored in S3-compatible buckets.

Connector Verification Scope

Cifru uses the same production functions in its automated connector tests through controlled transport fixtures. This proves request construction, authentication headers, catalog parsing, pagination, response parsing, row limiting, and error handling without requiring customer credentials. A fixture test is not falsely described as a live guarantee for every server version, proxy, firewall, plugin, or permission model.

Family	Automated end-to-end path	Additional evidence / remaining validation
SQL Server, PostgreSQL, MySQL, MariaDB, Oracle	Connection query, catalog, columns, generated read-only SELECT, preview, limits, destructive-SQL blocking, transient retry, and dialect-specific parsing run through the production pipeline.	A real read-only MariaDB/MySQL WooCommerce schema was checked independently with SHOW/INFORMATION_SCHEMA. Final device checks still require representative real servers for each engine and TLS mode.

Family	Automated end-to-end path	Additional evidence / remaining validation
HTTP feed, FTP, local file, S3-compatible	CSV, TSV, JSON, XML, XLSX, auth, sheet discovery, XML/JSON autodetection, FTP response parsing, signing, retry, and preview.	A public competitor-link XML feed was inspected as a real format example. FTP/S3 vendor/network combinations require release-device checks with authorized endpoints.
REST, GraphQL, OData	GET/query-only enforcement, auth, nested rows, pagination, filters/sort where standardized, GraphQL mutation/subscription blocking, and same-origin credential protection.	Custom APIs still require their real endpoint, response shape, and permissions to be tested by the user.
WooCommerce, Shopify, Magento, PrestaShop, OpenCart custom API	Official response shapes, auth method, known read objects, pagination, and bounded preview are tested through production connector code.	Store versions, plugins, scopes, custom fields, HPOS/legacy storage, and custom OpenCart routes require an authorized store test.

Sources Tab

Add Sources Area

Control	Type	Purpose
Add sources	Prominent menu button	Primary action for adding a source. Use this instead of relying on the small plus icon.
Plan usage rows	Status	Shows source, table/list, and data-link limits for Free, Starter, or Pro.
Source row status: OK	Badge	Cifru verified both the connection and a concrete readable structure/preview. A successful login alone is not shown as OK.
Source row status: Connected	Badge	Authentication or basic connectivity worked, but Cifru could not yet verify a table/schema, endpoint response, sheet, or feed row. Open the source and resolve the displayed next step before building.

Control	Type	Purpose
Source row status: Configure	Badge	The source exists but has not passed connection and data verification.
Source row status: Off	Badge	The source is disabled and should not be used during refresh.
Source row summary	Text	Shows connector type and a short connection summary, such as host:port / database, URL host, file name, or bucket/key.

Source Editor - Common Sections

Path: `Sources > select a source`.

Section	Control	Type	Purpose
Source	Name / alias	Editable text field	User-friendly identifier shown in Sources, every Build picker, diagnostics, and safe AI metadata. Use distinct aliases such as WooCommerce Romania, WooCommerce B2B, Production MySQL, or Reporting MySQL. When another source of the same type is added, Cifru proposes a numbered unique name that the user can replace.
Source	Kind	Picker/dropdown	Connector type. Changing it changes the source-specific fields.
Source	Enabled	Toggle	Turns the source on/off without deleting it.
Inline field hints / Ready to test	Validation hints	Status messages	Required database fields show their missing-field hints next to the field itself. Other connector hints appear in a short Complete fields section. Ready to test confirms that required fields are present.
Security	Warnings	Status rows	Warns about risky settings such as disabled TLS, HTTP, FTP, relaxed certificate validation, missing endpoints, high row limits.

Section	Control	Type	Purpose
Connection check	Check connection	Button	Saves the current source, tests login/connectivity, then automatically verifies the next real step: SQL catalog plus columns, or a bounded endpoint/feed/file preview. Only the complete path produces OK. A partial result is shown as Connected with a specific correction.
Connection check	Certificate hint	Status hint	Certificate failures explain that Trust server certificate should be used only for a private server controlled by the user. Certificate validation applies to SQL Server, PostgreSQL, MySQL, MariaDB, and Oracle.
Toolbar	Save	Button	Saves source settings and Keychain secrets, shows a visible confirmation near the top, then automatically runs connection plus structure/preview verification. Settings remain saved if verification fails, but the source does not show OK.

Database Source Fields

Applies to SQL Server, PostgreSQL, MySQL, MariaDB, and Oracle.

Field	Type	Notes for the AI assistant
Host	Editable text field	Server address, for example `server.example.com` or a VPN/LAN host.
Port	Numeric text field	Default examples: SQL Server 1433, PostgreSQL 5432, MySQL/MariaDB 3306, Oracle 1521. The user types the full port directly; do not tell them to use thousands separators.
Database / service	Editable text field	Database name, schema/service name, or service identifier depending on database type.
Username	Editable text field	Recommend a source-side read-only account.
Password	Secure field	Stored locally through the app's secret storage.

Field	Type	Notes for the AI assistant
TLS	Picker	SQL Server, PostgreSQL, MySQL, and MariaDB offer Prefer TLS, Require TLS, and No TLS. Prefer can fall back to plaintext if the server does not support TLS; use Require for sensitive/public-network traffic. Oracle exposes Require TLS or No TLS because its driver does not provide a truthful opportunistic mode.
Trust server certificate	Toggle	Certificate validation is enabled by default for all database engines. New SQL Server sources start with this toggle on for compatibility with private company certificates; new PostgreSQL/MySQL/MariaDB/Oracle sources start with it off. Turning it on keeps encryption but skips certificate-chain validation, so use it only for a controlled private server.

Feed, FTP, API, File, Storage, And Shop Fields

Connector	Available controls	How to guide the user
HTTP(S) feed	Optional base URL, Feed URL, File format picker: AUTO, CSV, TSV, JSON, XML, XLSX; Authentication picker; optional XML row name.	Use a direct structured-file URL. Cifru automatically recognizes common JSON envelopes such as data/results/items/records/rows/value and can autodetect a repeated XML row element when the default name does not match. Rows path and XML row name remain editable for unusual formats. Credentials are never forwarded to another origin.
FTP file	FTP host, Port, Username, Password, Remote file path, File format picker, optional XML row name.	Uses EPSV first and PASV fallback, reusing the configured host rather than trusting a passive-response host. Prefer HTTPS feeds because classic FTP is not encrypted. XLSX sheet browsing and automatic XML row detection use the same download path as the final list.
REST JSON	API address/base URL and Authentication picker: Automatic, None, HTTP Basic, Bearer header, API-key header, or Query token.	Choose endpoint paths while creating the Home button. Uses GET/read endpoints only, recognizes HTTP Link and common JSON next-page links, and follows pagination only on the exact same scheme/host/port. A query token is injected from Keychain when the request is sent, so the saved endpoint contains only its path and non-sensitive query options.

Connector	Available controls	How to guide the user
GraphQL	GraphQL API address and the same generic Authentication picker.	The Home button assistant query field becomes a GraphQL query editor. Only query operations are allowed. Authentication is applied by the shared request layer rather than embedded in the GraphQL text.
OData	OData API address and the same generic Authentication picker.	Choose Entity set/path while creating the Home button. Compatible filters and sorting are sent with standard `$filter`, `$orderby`, and `$top` query options. Existing public query parameters are preserved when authentication is added.
Local file	Choose file button, selected file name, File format picker, XML row name.	Imports CSV, TSV, JSON, XML, or XLSX from Files for local read-only use. For XLSX, use Build > Home screens > Choose source and table/path, then Browse sheets to choose which sheet becomes a Home button. Good for prototypes and static reports.
S3-compatible storage	Endpoint URL, Bucket, Object key/path, Region in Advanced mode, Access key, Secret key, File format picker, XML row name.	Use for structured files in object storage. Explain that the object should be a readable CSV/TSV/JSON/XML/XLSX file. For XLSX object storage, set Object key/path to the workbook, then use Browse sheets in the Home Button Assistant to choose the sheet.
WooCommerce	Store URL, Consumer key, Consumer secret.	Use read-only WooCommerce REST API credentials where possible. Known store data can be selected later.
Shopify	Shop domain, Admin API token, API version in Advanced mode.	Use a token with read scopes only. API version default is configured in the app.
Magento / Adobe Commerce	Store URL, Store code, Bearer token.	Use a token or integration with read permissions only.
PrestaShop	Store URL, masked API key, Response format picker, XML row name.	Use read webservice permissions. Cifru sends the key with HTTP Basic authentication and does not place it in the URL.

Connector	Available controls	How to guide the user
OpenCart / custom API	API URL, Token header in Advanced mode, Bearer token, masked API key.	Use custom read endpoints created by the store owner. If the API uses a header such as `X-API-Key`, enter that header in Advanced mode.

Generic HTTP Authentication

The HTTP(S) feed, REST JSON, GraphQL, OData, and generic custom-shop API connectors share one vendor-neutral authentication layer. Automatic keeps compatible legacy settings; None sends no credential; HTTP Basic uses username/password; Bearer header uses an Authorization-style token; API-key header lets the user choose a valid header name; Query token lets the user choose the parameter name. Secret values are stored in Keychain and are not serialized into workspace JSON, copied into endpoint fields, shown in query transparency, or forwarded to a different scheme/host/port. Query-token mode builds the final URL with `URLComponents`, replaces a duplicate secret parameter, and keeps all other non-sensitive query items intact.

Home Button Assistant

Path: `Build > Home screens > Choose source and table/path`. This is the easiest path for non-programmers. The source editor is only for connection settings and connection testing; tables, views, endpoints, sheets, and feed paths are chosen here while creating a Home button.

Section	Controls	Purpose
Progress	Check connection, Choose data, Choose fields, Create Home button.	Shows where the user is in the source-to-screen setup flow. Details and cross-source values are configured afterward in the same screen Studio.
Starting point	Name, Type.	Confirms which source is being configured.
1. Check connection	Check connection button and status row.	Tests the source before browsing or previewing. The next steps are disabled until this succeeds. If the source was already tested from Sources, this step starts as done.

Section	Controls	Purpose
2. Choose data	Browse button, searchable modal for tables/endpoints/sheets, object path text field, optional Rows path field for nested JSON/API responses, GraphQL query editor when applicable, Save selection button, Save selection & preview rows button.	Chooses the table, endpoint, feed path, entity set, GraphQL query, known store data, or XLSX sheet. Browse opens a popup/modal over the assistant, not a huge inline list. For the default SQL schema, a canonical name such as `dbo.accesex_arhiva_view` is shown simply as `accesex_arhiva_view`; Cifru keeps the complete canonical name internally. Non-default schemas remain visible when needed to distinguish objects. Selecting a row fills the table/path field directly so the user does not need to type technical names manually. Cifru remembers the last saved selection for that source.
3. Choose fields	Field count, visible-field count, and a searchable field picker based on the concrete source schema. Basic mode asks for selected fields, friendly labels, and card/detail visibility; internal names, types, presentation, optional semantic meaning, and the stable unique-row field remain in Advanced mode. Test and preview is the final step after the fields are chosen.	Maps real source fields into mobile-friendly labels without asking the user to remember technical column names. A semantic field meaning helps formatting and later suggestions; it does not connect this list to another source.
4. Add to Home	Button/list name text field, summary labels, Add button to Home button.	Creates the list and the Home button in one guided step.

> AI guidance pattern: tell the user to first test the source in Sources, then go to **Build > Home screens > Choose source and table/path**. After the user chooses or types a table/path/sheet, tell them to tap **Save selection & preview rows**. Fields are shown only after a preview for the current selection and Rows path, so Cifru does not reuse mappings from an older table, endpoint, sheet, or nested JSON path.
>
> One configured source can produce multiple lists/Home screens. After creating the first Home screen, the user can tap **Add another table/list from this source**, choose another SQL table, endpoint, or XLSX sheet, preview it, choose fields, and create another Home screen without duplicating the source connection.
>
> Builder concept: a Source is only the connection. A List, also shown as Data to read, chooses what to read from that source. A Home screen/button chooses how that list appears as mobile cards. Card fields from other lists and row Details are configured on Home screens/buttons, not inside the source connection.

Build Tab

Path: `Build`. This tab is for configuring the workspace after sources exist.

If there is no successfully tested source, Build shows a prominent warning and a highlighted Go to Sources button. The Sources tab also shows a badge until at least one enabled source has a successful connection test. This is intentional: users should configure and test a source before building lists, Home buttons, Details, or widgets. Once a source is OK, Build shows the guided Create a screen or widget action, where the user can choose a recipe, source, and table/path, preview rows, and create the first Home screen without guessing which technical area to open.

In Cifru builder editors, **Save** stores the current settings, shows a short visible confirmation at the top of the editor, and keeps the user on the same screen. Use the Back button when finished editing. This lets the user save, preview, test refresh, and continue adjusting without being sent back to the previous category.

Build uses a hybrid organization. Basic mode starts with a compact tree of what the user actually sees: each Home screen is a parent row, its Details screens are nested children, and cross-source enrichments are shown under the screen where their values appear. Tapping a normal screen opens one Studio with five contextual tabs: **Card**, **Data**, **Filters**, **Actions**, and **Preview**. Card is the default tab so the first view matches what the user recognizes on Home. Its visual layout editor has separate **Compact card** and **Full detail** surfaces. A user can enable or hide fields, rename labels, choose text/link/image presentation, promote totals or statuses as highlights, group

information, enter Edit mode, and drag fields into a page-specific order. A recommended-layout action keeps useful business fields prominent and moves technical identifiers out of the compact card. Guided shortcuts may open Data or Actions directly when that is the next unfinished step. This removes the need to understand three parallel technical areas before making a useful screen.

Each main row in **Your screens** has an up/down menu. Reordering a main screen moves its complete navigation tree as one unit, including nested Details screens, sub-buttons, linked/enrichment data, and saved relation references. The resulting main-screen order is also the Home order. Child rows are intentionally not reordered independently from this overview because their presentation belongs to the parent screen and is configured in Studio.

The **Create a screen or widget** action offers three guided recipes: **Home button with list**, **Home button with Details**, and **Widget with total, current value, or ranking**. A new Home screen follows one uniform sequence: name the screen, choose the configured source, then open **Choose table/path and fields**. This third step is the exact same data editor used later in **Studio > Data**, so table/view/sheet/endpoint/feed discovery, field selection, preview, and Pro Advanced custom read-only queries are never duplicated in a second setup surface. After the data is verified, the user continues in the same Studio. **Home button with Details** opens Studio at **Actions**, where a reusable child list becomes a button on every matching card; it is not shown as an unrelated second Home button. Cross-source values are added under Studio Data or Actions through a common field rather than through a separate builder recipe. The AI Configurator remains a separate visible toolbar action in the upper-right of Build.

While a new screen is still a draft, Studio shows a short completion checklist: readable table/path or tested custom query, real fields selected, card layout completed, and, for the Details recipe, one successfully tested per-card Details action. The final Save action remains unavailable until the required steps are complete. Cancel removes only temporary draft elements created by that flow.

Creating and editing a screen is transactional from the user's perspective. Canceling a new draft removes its temporary screen and any list created exclusively for that draft. If the user chose an existing list, canceling removes only the new screen and preserves the reused list and every other screen using it. Leaving or discarding unsaved changes on an existing screen restores the saved screen and never deletes it; deletion is available only through a separate destructive action and confirmation. Saving keeps the screen and continues with the same Studio controls.

Every Studio entry begins with a **What you are editing** identity card. It distinguishes a main Home button, a Details/sub-button screen (including its parent and relation), a sub-button destination, linked/enrichment data, and a new temporary Home draft. When Studio Data opens the shared list editor, the editor repeats the linked-data identity so the user does not mistake source/list settings for the main Home card layout. Each context edits its own saved page presentation and actions while reusing the underlying read-only list where appropriate.

Advanced mode reveals the underlying **Data library**, **Relations map**, and **Common fields**. These views edit the same lists and links used by Studio; no setting or capability is removed. Lists without a Home screen remain available in the Advanced data library because they may feed a widget, Details action, or cross-source enrichment. With an active Pro subscription, SQL-backed lists also expose an optional custom read-only query editor. The editor accepts a complex SELECT/WITH block with CTEs, joins, unions, cases, windows and aggregations against the selected connection. With explicit approval, its AI helper first researches official public manuals, uses bounded object and generic field terms to inspect the live catalog, and generates only from the exact discovered schema. A contradictory documentation plan degrades to schema-first generation. An independent coverage critic may request one bounded correction, but it cannot override exact local validation and the bounded live read. The current SQL draft remains visible for inspection and copying; Save stays disabled until that exact text passes the read-only test. Authentication, permission, network, TLS and timeout failures are never sent back to AI as repair tasks. The same editor exposes a visible Test read/Test and preview action, and a successful custom-query test is carried back into the new-screen checklist. Basic mode never requires SQL.

Build screen	What it configures	When to use it
Sources	Connection profiles and credentials.	When the source address, credentials, TLS, token, or feed file changes.
Compact screen tree	Home screens, nested Details screens, and enrichments in the same hierarchy the user will navigate. The up/down menu moves one complete main-screen tree and changes the Home order.	The normal starting point for opening, reviewing, or ordering existing dashboard screens.
Page Studio: Card (default)	Home title, subtitle, icon, title/subtitle/badge, recommended layout, and separate draggable field order/visibility/labels/presentation for the compact card and full detail. Full detail fields can be highlights or grouped information. Add a field opens the same calculated-field chooser used by Data and reveals the result only on the current layout.	The default entry point; use it when changing how each row and its complete workspace look to the user. Layout belongs to the screen, so two screens may present the same underlying list differently.
Page Studio: Data	Source list, concrete table/path/sheet, cache/live mode, row limit, date period, source fields, local calculated fields, mappings, refresh, and search.	When changing what rows or row-level values the selected screen reads. A guided setup shortcut may open this tab directly.

Build screen	What it configures	When to use it
Page Studio: Filters	Opening filters, date period, sorting, and list-navigation behavior.	When choosing which rows open first and in what order.
Page Studio: Actions	Buttons and related sections on every card, including guided parent-to-child Details. Each related list can be inline, collapsible, a tab, or a separate screen, with card, compact-row, or table presentation.	When a row must show matching lines/items or perform another already implemented read-only action.
Page Studio: Preview	The actual mobile compact-card preview, the composite full-detail workspace, and a summary of related sections and enrichments.	Before saving or after changing the selected screen.
Advanced data library	All lists, including lists without a Home screen.	Technical inspection and reuse of data that feeds widgets, Details, or enrichments.
Advanced relations map	Parent-child links between lists.	Technical inspection or editing of relationships; normal users can create Details from Studio.
Advanced field meanings	Optional semantic labels used for formatting and later suggestions.	When a technical column should be recognized as a date, document ID, EAN, amount, or another familiar meaning. This annotation is not itself a cross-source link.
Advanced read-only query (Pro)	One validated SQL execution block for the current database list, plus an optional AI drafting helper.	When an expert needs CTEs, nested SELECTs, joins, unions, CASE expressions, aggregates, or window functions that the guided builder cannot express.

Build screen	What it configures	When to use it
Widgets	Counts, sums, averages, min/max, top lists, filters, sort, external widget snapshot.	When the user wants dashboard summaries or Home Screen / Watch / CarPlay widgets.
Subscription	Free, Starter, Pro purchase/restore and feature limits.	When feature gating blocks sources, links, or advanced widgets.

Lists Editor

Path: `Build > Lists > select list`.

Section	Control	Type	Purpose
List	Name	Text field	User-facing list name.
List	Reads from	Picker	Selects the source used by this list.
List	Loading	Picker: Cache or Live query	Cache stores rows locally. Live query reads fresh data when needed.
List	Limit number of rows	Toggle	On by default to protect mobile performance and source load. Turn it Off when a source-side filter, such as one calendar day, should return every matching row.
List	Max rows	Direct numeric field, 1 to 50000	Limits rows only while Limit number of rows is On. The value can be typed directly; it is not a +/- stepper.
List	Unique row field	Text field + picker in Advanced/when set	Optional stable key for row identity.
What to read	Choose/change table, view, sheet, endpoint or path	Searchable source-specific selector	Discovers the real selectable objects from the configured source. The selected canonical value is saved even when a default SQL schema prefix is hidden from the friendly title.

Section	Control	Type	Purpose
What to read	Custom read-only query	Pro Advanced multi-line editor for SQL database sources	Uses the selected database connection and supports one complex block beginning with `SELECT` or `WITH`. The exact query must pass local validation and a bounded test before Save is enabled. Selecting a normal table/view later exits custom-query mode.
What to read	GraphQL query	Multi-line editor	Read-only GraphQL query.
What to read	API/feed path, Object key/path, Entity set/path	Text field	Path relative to the source base, object key, or OData entity.
What to read	Known store data	Picker for ecommerce sources	Selects friendly known store endpoints when available.
What to read	Rows path in JSON/response	Text field for feeds/APIs/files/storage when useful; also visible in Advanced mode	Optional path to the array of rows, for example `data.items`, `data.orders.nodes`, or `value`. OData uses `value` by default.
Refresh	Auto refresh cache	Toggle	Turns timer-based refresh on/off for cached lists. While Cifru is active, one app-level scheduler checks due lists and widgets about every 30 seconds, independently of the currently visible screen. Returning to the app also performs an immediate due-cache check.

Section	Control	Type	Purpose
Refresh	Every X min	Stepper 1 to 1440	Requested refresh interval for cached rows. Due lists and widgets share a serial, deduplicated queue that alternates both work types and prioritizes relatively overdue items, so a large list installation cannot indefinitely keep widgets behind every list. The editor shows whether work is due, running, queued, or waiting for automatic retry. Foreground checks run promptly; background execution is best-effort because iOS decides when the app receives background time, so the interval is a minimum due time rather than an exact background alarm.
Common fields	Detect common fields	Button	Attempts to map fields to shared meanings such as Invoice ID, Customer ID, EAN/Barcode.
Preview before saving	Preview rows	Read-only preview	Shows cached rows so the user can check fields before saving. Cifru prefers a meaningful business value as the preview subject, omits technical ID/UUID/serial fields from the summary, and uses Sample row 1, Sample row 2, and so on when no friendly value is available.
Fields to show	Detect and choose fields / Choose from detected fields	Searchable multi-select modal	Reads actual SQL columns, API/feed keys, XML elements, CSV/TSV headers, or XLSX sheet columns. Tap a field to add/remove it; selected fields are checked.
Fields to show	Detect and choose fields	Button	Always re-runs source-specific schema detection, then opens the searchable checkbox picker. After a table, endpoint, feed, row path, or sheet changes, compatible selections are preserved and fields that do not exist in the new object are removed.

Section	Control	Type	Purpose
Fields to show	Add fields...	Guided choice after the detected fields and in the list editor	Separates two different operations: Calculate a field from this row, or Bring fields from another list or source. The custom SQL editor is not duplicated here and remains under What to read.
Fields to show	Calculate a field from this row	Universal guided formula editor	Creates one local numeric field from two fields or from one field and a fixed value. Supported operations are add, subtract, multiply, divide, and percentage, with 0-6 decimal places and an explicit missing-value rule. A live local preview is shown when a numeric row is available. Division by zero and nonnumeric operands produce an empty result. Formulas are declarative, execute locally after a read, never modify the source, work for SQL/APIs/feeds/files, and consume no additional source/list/relation plan allowance.
Fields to show	Bring fields from another list or source	Four-step read-only relation assistant	Choose an existing list or another concrete table/sheet/path from any accessible source, choose the field on the current card and the matching field in the other data, select returned fields, then review matching analysis. Cifru checks bounded distinct values across representative source data rather than trusting only the first page. Repeating the same field pair reuses the existing relation and merges selected card fields.

Section	Control	Type	Purpose
Fields to show	Created fields	Formula rows marked fx, with Edit/Delete and preview	The formula belongs to the list, but new calculated mappings default to hidden globally. When created from one Home/Details/sub-button screen, only that screen's selected card/full-detail layout is enabled automatically. Other screens that reuse the same list may opt in independently.
Fields to show	Manually add a source field	Advanced guided sheet below detection	Maps an existing field returned by the selected source when automatic discovery cannot expose it. The user may choose a detected field or type the exact source name, review a local sample, set the friendly label, type, link/image presentation, list/detail visibility, and optional common-field meaning. Cifru proposes a stable internal key and blocks duplicates, empty names, and fields hidden from both list and details. Use Calculate a field from this row when the value does not exist in the source.
Fields to show	Original column	Read-only caption in Basic; editable in Advanced	Exact source column/key name. Basic users select it instead of typing it.
Fields to show	Internal field name	Advanced text field	Stable internal key used by buttons, filters, widgets, and links.
Fields to show	Display label	Text field	Friendly label shown to the user.
Fields to show	Type	Advanced picker: Text, Number, Date	Detected automatically; Advanced mode can correct it for sorting/filtering and numeric widgets.
Fields to show	Display as	Advanced picker: Automatic, Text, Link, Image	Controls presentation without changing how the field is matched. Automatic recognizes web/mail links and image-like fields.

Section	Control	Type	Purpose
Fields to show	Common field	Advanced picker	Marks the meaning of the field for cross-source matching. Cifru proposes common roles where possible.
Fields to show	Show in list / Show in details	Toggles	Controls where the field appears.
Fields to show	Local sample	Read-only privacy-sensitive text	Shows a value from the bounded local preview to help identify a field. This displayed preview value is not sent directly to external AI. A separate optional repair-only identifier sample follows the strict limits described under AI Configurator privacy.
Search uses	Field names / toggles	Advanced controls	Controls which fields are searched inside list browsing.
Toolbar	Test refresh	Button	Saves the current list settings first, then pulls a preview from the source.
Toolbar	Save	Button	Saves the list configuration.

Advanced Read-Only Queries (Pro)

Paths: `Build > Create a screen or widget > Home button > name > source > Choose table/path and fields > Custom read-only query`, or `Build > select a screen or Details list > Data > Choose table/path and fields > Custom read-only query`. Both paths open the same editor and use the same validation/test rules. This expert feature is available only while Pro and Advanced mode are active. It works for a new or existing Home list and for a list opened by a per-card action.

Area	Behavior	Safety rule
Query shape	One read-only execution block may contain CTEs, nested SELECTs, joins, unions, CASE expressions, aggregates, and window functions.	Independent semicolon-separated statements are rejected. Studio controls final sorting, filters, and row limits, so a top-level custom `ORDER BY` is not accepted.

Area	Behavior	Safety rule
Query block	**Copy SQL** copies the current editor text and confirms with **Copied**.	Copying is a local, explicit clipboard action. It does not test, execute, save, or send the query.
Database scope	Direct SQL joins can use tables/views reachable through the currently selected database connection.	A second configured connector is not exposed as a fake SQL table. SQL Server, Oracle, MySQL, a feed, API, file, or ecommerce source configured separately is combined afterward through Cifru's existing local relation/enrichment controls.
Source aliases	The editor and AI helper show safe source IDs/display aliases and whether each source is directly queryable or must be mapped locally.	Aliases never contain credentials, rows, connection strings, private URLs, or query results. Display names are treated as metadata, not instructions.
Validation	Cifru validates locally before any database request and automatically runs a preview limited to 20 rows through the same production database runner used by the final screen.	The preview starts with a neutral field/sort configuration. Fields and sorting from an older query are never added around a newly generated query. INSERT, UPDATE, DELETE, MERGE, DROP, ALTER, CREATE, TRUNCATE, EXEC/EXECUTE, GRANT/REVOKE, SELECT INTO, locks, external/linked execution, deliberate waits/sleeps, and other write/admin behavior are blocked.

Area	Behavior	Safety rule
AI helper	An optional collapsed assistant can draft the query from a plain-language request and the safe discovered schema. With explicit approval, an external provider first consults official public manuals and returns documentation evidence plus a vendor-neutral semantic contract: separate business roles, the owning role of each requested output, direct or documented formatted-identifier relations, final row grain, qualifying roles, and distinct top-N identity/order rules. Cifru then expands the live catalog with exact object candidates and generic concept field aliases (including common multilingual line/detail and creator/date wording) before a separate strict-JSON query draft is generated.	The exact locally discovered objects, fields, and field types remain authoritative. Catalog search reads metadata only. During schema-first repair, the optional Minimal mapping samples control may permit the already disclosed identifier-only sample; Cifru never sends full rows, names, invoice contents, prices, credentials, connection strings, or private URLs. A contradictory research contract receives one correction and then degrades to schema-first generation. The independent draft-coverage critic receives at most one repair attempt; remaining critic uncertainty becomes a visible warning and cannot replace deterministic local validation or the bounded live read. A draft stays visible but unsaveable after a failed check. The exact query must return a non-empty preview of at most 20 rows before Save is enabled, and a successful test is propagated to the screen-creation checklist.
Changed query fields	After a successful test, Cifru proposes fields from the actual returned columns. Matching display labels/settings may be preserved.	Source columns, search fields, primary keys, and sort rules that belonged only to the previous query are removed, preventing a valid new SELECT from failing because an obsolete outer field was reused. Local calculated-field definitions remain separate; a formula returns empty until its selected input fields exist again.
Errors	Database diagnostics such as missing column, missing object, wrong dialect syntax, permission, TLS, network, or timeout are translated into an actionable message.	Cifru does not replace a useful driver/server diagnostic with the generic iOS text “The operation couldn’t be completed.” Secrets are not included in the displayed diagnostic.

Area	Behavior	Safety rule
Runtime protection	The normal source timeout, editable data-safety timeout, cancellation, adaptive row protection, Studio filters, and optional row limit remain active.	Use a source-side read-only database account as an additional permission boundary. Cifru representatives are not responsible for the correctness, cost, business meaning, or source permissions of user-entered queries.
Subscription downgrade	The saved query configuration remains on the device so the user's work is not deleted.	Configuration, discovery, testing, and execution are disabled until Pro access is restored.

> **Calculated-field scope:** formulas are evaluated locally after Cifru receives each bounded row. They can be displayed, searched, filtered locally, exported, used in Details/sub-buttons, and selected by local widgets after refresh. They cannot change which rows a source returns before its source-side filter/sort/limit. Use the existing validated custom SQL query only when a database must filter, rank, group, or limit by a derived expression before sending rows to the device.

Common Fields

Path: `Build > Common fields`. Common fields make cross-source mapping user-friendly. A common field is a semantic annotation and normalization rule, not a relation by itself. Under every common field, Cifru discreetly reports whether it is unused or which list mappings and saved relations currently use it.

Control	Type	Purpose
Add common field	Button	Creates a new meaning such as "Project ID" or "Serial number".
Restore defaults	Button	Restores built-in common fields.
Name users see	Text field	Friendly label for the meaning.
Type	Picker: Text, Number, Date	How values should be compared.

Control	Type	Purpose
Alternative names	Comma-separated text field	Synonyms used to detect mappings, such as `ean, barcode, gtin`.
Trim spaces at ends	Toggle	Removes leading/trailing spaces before comparing.
Text case	Picker: Keep case, Lowercase, Uppercase	Normalizes text before matching.
Remove all spaces	Toggle	Useful for codes with inconsistent spacing.
Remove punctuation	Toggle	Useful for formatted IDs or phone-like codes.
Remove leading zeros	Toggle	Useful when one system stores `000123` and another stores `123`.
Use for link suggestions	Toggle	Allows this common field to be used for automatic Details and cross-source link suggestions.
Workspace usage	Read-only status	Shows the number and names of lists whose mappings use this meaning and how many relations depend on those concrete mapped fields.

Data Links / Details

Path: `Build > screen > Studio > Actions` for a Details section/shortcut on every card, or `Studio > Data` for extra values shown on the same card. Advanced mode also exposes `Build > Link data` as a technical relation map. A data link takes one concrete value from the opened parent row and reads only child rows whose configured field has the same value. It is read-only and can power either a one-to-many Details button/section or a card enrichment. The map now states this role, both concrete list/field endpoints, whether the relation is available under the active plan, and exactly which screen currently uses it; unused saved relations are identified as such.

Control	Type	Purpose
Name	Text field	Friendly relation name, such as Invoice details.

Control	Type	Purpose
Main list	Picker	Parent list opened from Home, such as Customer invoices.
Field from opened row	Text field + picker	Parent key, such as `invoice_id`.
Details list	Picker	Child list, such as Invoice lines.
Common field from details	Text field + picker	Child key, such as `invoice_id`.
Suggest common fields	Button	Uses common-field mappings and synonyms to propose parent/child keys.
Test Details now	Button	Uses one real card already loaded on the device and performs the same read-only lookup that the Details button will use. It reports loaded rows, no match, or a source/configuration error before the user saves.
Preview before saving	Matching rate	Shows percent and matched/total count when both lists already have rows. The direct Test Details button is the more reliable check when the child list is not cached yet.
Example	Read-only label	Shows a sample match such as `invoice_id 123 -> 4 matching rows`.
Save	Button	Saves the data link.

Card Fields From Other Lists

Path: `Build > Home screens / buttons > select screen > Fields from other lists`. This is the simpler choice when the user wants a few values from one matching row to appear on the same card.

Control	Type	Purpose
Add fields from another list	Button	Creates a same-card data link, subject to the same subscription limits as other data links. The same guided flow is also available at the bottom of the detected-field picker in Studio > Data, so a non-technical user can start from a real current field instead of navigating to a separate technical area.
Name	Text field	Friendly name for this enrichment.
Read fields from	Picker	The other list that contains the values to display.
Common field on this card	Text field + picker	Field from the current card, for example `customer_id`, `invoice_id`, `ean`, or `sku`.
Common field in the other list	Text field + picker	Matching field in the other list.
Suggest common fields	Button	Uses common-field mappings to suggest the matching pair.
Fields to display	Field picker + label text field + Display as picker	Selects which values from the other list appear on the current card and whether each value is rendered automatically, as text, as a link, or as an image.
Preview before saving	Matching rate + sample values	Lets the user verify that the chosen fields really match before saving.

> AI guidance pattern: if the user says "put customer phone/product name/status on the same invoice/order/card", guide them to Fields from other lists. If the user says "open all invoice lines/tasks/items/details", guide them to Home screens / buttons > Add guided Details.
>
> Cifru first tries a narrow read-only lookup. If the two sources need common-field normalization, for example EAN `0594123` versus `594123`, it can use a bounded fallback scan and compare normalized values locally. A custom API that does not support filters may therefore need an adequate Max rows setting or a server endpoint designed for filtered reads.

Home Screens / Buttons Editor

Path: `Build > Home screens / buttons > select screen`. Home screens open lists from the app Home screen and define the mobile card experience.

Use this editor for the normal master-detail flow. For example, Supplier invoices is the main list/card screen, and a Details shortcut focuses invoice lines by matching `id_document` from the opened invoice to `id_document` in the invoice-lines list. The invoice header, highlights, metadata, and lines stay in one document workspace. The details list can come from the same source or a different source.

Mental model for non-programmers: the list/table is what Cifru reads; the Home button is how the user opens that list on the phone; Details is a related section for matching child rows; Fields from other lists adds a few values from one matching row onto the same card.

Section	Control	Type	Purpose
Home screen/button	Title	Text field	Main label on the Home card.
Home screen/button	Subtitle	Text field	Secondary text on the Home card.

Section	Control	Type	Purpose
Home screen/button	Icon	Visual picker	Selects a familiar built-in icon for documents, orders, products, customers, reports, links, calendar, archive, and other common uses. Existing custom SF Symbol values are preserved as the current icon, but normal users do not need to type an internal icon name.
Home screen/button	Show on Home	Toggle	Hides/shows the button. Hidden pages can still be used as detail screens.
Home screen/button	List to open	Picker	Selects which list opens when tapped.
Screen structure	Add guided Details	Button	Opens the Details Assistant. The assistant asks where the details rows come from, which field on the opened card matches which field in the details list, shows matching-rate preview, then saves the Details button on every row card. For invoices, use the invoice header list as the parent and the invoice-lines list as the details list.
Data loading	Loading	Picker: Cache or Live query	Edits the linked list loading mode without hunting in Lists.
Data loading	Auto refresh cache	Toggle	Turns automatic cache refresh on/off for the linked list.
Data loading	Every X min	Stepper	Refresh interval for the linked list.
Data loading	Limit number of rows	Toggle	Controls whether the linked list has a final row cap.
Data loading	Max rows	Direct numeric field	Editable row cap shown only when limiting is enabled. For generated SQL table/view reads, date filters and saved sort rules are applied before this limit.
Data loading	Refresh this list now	Button	Refreshes this list immediately.

Section	Control	Type	Purpose
Preview before saving	Home card + compact row card + row-interior preview	Read-only preview	Shows the same visual hierarchy used at runtime before the screen is saved.
Row display	Row title field, subtitle field, badge field	Text fields + pickers	Selects which columns appear as the row title, subtitle, and badge.
Row display	Compact card / Full detail selector, Recommended layout, field visibility, labels, presentation, detail role/group, and drag order	Studio > Card	Builds a readable compact card and a separately ordered composite detail. Full detail can promote totals/statuses as highlights and group metadata. Technical fields can stay available inside while remaining hidden on the compact card.
Row display	Rows per page	Stepper 10 to 500	Controls paging to avoid huge mobile lists.
Filter before opening	Add opening filter	Button	Adds a prompt/dropdown-like filter before the list opens, for example Store, Year, Status. When possible, Cifru preselects likely fields such as gestiune, magazin, store, location, warehouse, branch, or depot.
Filter before opening	Title, Field, Type, Allow All option	Text fields, picker, toggle	Defines the opening filter. Type can be Text, Number, or Date. Allow All lets the user explicitly choose Show all, but the list does not silently open on All before the user confirms.
Filter before opening	Require filter selection when opening	Toggle	When On, the Home screen waits for the user to choose a filter value or All before reading the list. When Off, the list opens immediately and the same filters remain available as optional controls above the rows.

Section	Control	Type	Purpose
Data period	Date field + Period	Pickers, calendar, direct numeric field	Available only when the selected list contains a field detected or manually marked as Date. It applies the current Day, Week, Month, or Year, as well as Yesterday, Last X days, or one specific calendar day before the row cap. If no Date field exists, Cifru shows a setup hint instead of an inapplicable filter. Turn the cap Off to retrieve every row matching that period. Last X days is typed directly.
Sort	Add sort	Button	Adds sorting by one or more fields. When possible, Cifru preselects a Date-like field and Descending direction, which is usually right for latest invoices/orders first. For direct SQL lists, this order is used before Max rows when refreshing.
Sort	Field, Type, Direction	Text field/picker, type picker, direction picker	Direction is Ascending or Descending. Set Type correctly for date/number sorting.
Fields from other lists	Add fields from another list	Button	Adds selected values from one matching row in another list directly onto the current card.
Fields from other lists	Name, Read fields from, common field on this card, common field in the other list, Suggest common fields, Fields to display, Matching-rate preview	Text fields, pickers, button, preview	Use for 1-to-1 enrichment such as invoice card + customer phone, order line + product name, task card + project owner. Use Details and row actions for 1-to-many data.

Section	Control	Type	Purpose
Details and row actions	Add Details / row action	Menu	Guided Details opens the Details Assistant. Each related list can appear inline, collapsible, as a tab, or on a separate screen; rows can be cards, compact rows, or a table, with a configurable preview limit and count. Advanced actions can still open another list or open a URL from the current row. A shortcut under the compact card opens the same composite workspace and focuses the chosen related section.
Toolbar	Save	Button	Saves the Home button.

Details Assistant And Card Actions

Path: `Build > select a Home screen > Studio > Actions > Add Details`.

Use the Details Assistant for the normal non-technical flow. The user should think: "When I open one invoice/order/project card, keep its header visible and show me the matching rows from this other table/list below it." The assistant keeps all required context visible: main list, details list/table, field from opened card, field in details, matching rate, presentation mode, row style, preview limit, and an example before saving.

Example explanation shown in-app: if the opened invoice card has `id_document = 123`, Details should show only rows from the invoice-lines table where `id_document = 123`. This prevents users from confusing Details with a separate top-level Home button.

Assistant step	Controls	Purpose
What this builds	Read-only explanation	Explains that Cifru will add a Details button to each card in the current Home screen.

Assistant step	Controls	Purpose
1. Where do Details come from?	Main list label, Details list/table picker, Create a new details list button	Selects an existing child list or creates one without leaving the Actions flow. The creation assistant browses a real configured source and lets the user choose its table, view, endpoint, sheet, feed, or path. It saves a reusable details list without adding a separate Home button, then returns it selected here. The child data can come from the same source or another source if the plan allows it.
2. Which field connects them?	Field from opened card picker, Field in details picker, Suggest fields automatically button	Defines the parent-child match without writing SQL. Basic mode shows friendly field labels; Advanced mode also shows technical keys. Example: Customer invoices.invoice_id equals Invoice lines.invoice_id.
3. Preview and save	Section text, inline/collapsible/tab/separate presentation, card/compact/table row style, preview limit, count/expanded options, matching-rate preview, Test Details now button, result/diagnostic, Save Details button	Tests the relationship against the source before saving and configures how it appears in the composite record. Cifru automatically resolves friendly labels, source column names, and internal mapped keys to the same field. If no rows match, the result explains which value and fields were checked.

> Runtime behavior: opening a record shows one composite workspace with the selected parent header, automatic or selected highlights, grouped information, and related sections. Each related section has its own visible loading/error/Retry state, so one failing child source does not erase the parent or other sections. A child row opens an inspector sheet that retains the parent context. For SQL databases, the equality condition is sent to the database before TOP/LIMIT. For feeds and files, Cifru filters parsed rows locally before applying the configured row limit. For paginated ecommerce/API connectors that do not expose a portable server filter, Cifru scans a bounded larger window, normalizes configured Common fields such as EAN/SKU, filters it, and only then applies the visible list limit. This reduces false empty Details/enrichment results caused by spaces, punctuation, leading zeros, or later API pages without requesting an unbounded dataset.
>
> Mapped-field safety: a relation remains valid when a source column, internal Cifru key, or friendly display label refers to the same field. After SQL schema refresh, invalid assumed fields are removed and page/relation references are repaired where an unambiguous mapped field exists.

Advanced path: `Build > select a Home screen > Studio > Actions > Open another list / Open URL`.

Row button type	Controls	Use case
Guided Details / Show matching details	Details Assistant with button text, Details list picker, parent field, child field, Suggest fields automatically, matching-rate preview.	Invoice -> invoice lines, order -> order items, project -> tasks, product -> stock by location. For MAXapp-style invoices, use `id_document` on both the invoice list and the invoice-lines list when that is the shared key.
Open another list	Title, Type picker, target List picker.	Simple navigation to another configured list.
Open URL	Title, Type picker, URL field text/picker.	Open a URL stored in a row, such as product page, document link, tracking link.

Widgets

Path: `Build > Widgets`. Basic mode exposes guided Count, Sum, and Current value controls, field pickers, date-period presets, and simple conditions. Current value is connector-neutral: it can

show the latest or otherwise sorted value from a database, API, feed, file, ecommerce source, or another configured read-only list, such as a sensor reading, ticket status, exchange value, or latest order state. Advanced mode additionally exposes Average, Minimum, Maximum, Top list, manual technical keys/types/operators, sort internals, and the explicit Cache versus Live-query choice.

For SQL Server, PostgreSQL, MySQL, MariaDB, and Oracle metric widgets, Cifru sends one generated read-only aggregate query such as `SUM`, `COUNT`, `AVG`, `MIN`, or `MAX` to the selected concrete table/view. The result is therefore not calculated from only the first 100 displayed rows. Date presets are applied at the source before aggregation. Feed, file, API, and commerce widgets use the compatible connector's bounded filtered read path.

Widget kind	Required fields	Notes
Count	List, optional filters.	Counts matching rows.
Sum	List, value field, optional filters.	Sums numeric values.
Current value	List, displayed field, optional date/sort field and filters.	Shows one exact text or numeric value from the first row after the configured source-side filters and descending/ascending sort. It is not limited to medical or sensor data.
Average	List, value field, optional filters.	Pro/advanced dashboard widget.
Minimum	List, value field, optional filters.	Pro/advanced dashboard widget.
Maximum	List, value field, optional filters.	Pro/advanced dashboard widget.
Top list	List, value/title/subtitle fields, limit, sort rules.	Shows a compact ranked list. Pro/advanced dashboard widget.

Widget Editor Controls

Section	Control	Type	Purpose
Preview before saving	Widget card preview and matching rows count	Read-only preview	Check the result before saving.
Widget	Title	Text field	Widget label.
Widget	Kind	Picker	Select Count, Sum, Current value, Average, Minimum, Maximum, or Top list according to plan/mode.
Widget	Data used	Picker	Source data list for the widget calculation.
Period and filters	Use	Picker: Inherit list filters / Custom widget settings	When one compatible Home list already has a period or fixed filters, Cifru can inherit those settings. The editor explicitly shows **Inherited list: [name]**, the inherited period, and the number of inherited fixed conditions. Choose Custom when the widget needs an independent period. Interactive opening filters are never inherited because a Lock Screen or Watch widget cannot ask the user for a choice.
Widget	Show on iPhone Home/Lock Screens, Apple Watch face, and CarPlay	Toggle	Adds this widget to the external local snapshot. When Use as app has a client logo, Cifru includes a small resized local copy for supported iPhone/CarPlay widget families and otherwise uses the Cifru/system icon. On watchOS 10 or later, Cifru publishes one preconfigured complication recommendation for every selected widget, using its configured title rather than numbered slots. watchOS 9 uses the first selected widget as a compatibility fallback. CarPlay uses the small widget where supported by iOS.

Section	Control	Type	Purpose
Widget	Freshness text	Read-only snapshot status	The external widget shows when its local snapshot was updated and uses the configured metric icon. It has no manual refresh symbol or action. Cifru refreshes enabled caches through its serial in-app/background pipeline when the operating system grants execution time, publishes a new App Group snapshot, and asks WidgetKit to reload. The extension never opens the source or credentials itself, and exact background timing is not guaranteed.
Widget	Value field	Friendly field picker; manual key in Advanced	Numeric field for Sum/Average/Minimum/Maximum, exact displayed field for Current value, or ranking value for Top list. Field references are reconciled again after SQL schema refresh.
Widget	Title field / Subtitle field	Friendly field pickers; manual keys in Advanced	Used by Top list rows. Date values use the app/phone date format instead of raw source text.
Widget	Limit	Stepper 1 to 50	Number of rows shown in Top list.
Data period	Date field, Period	Pickers: None, Current day/week/month/year, Yesterday, Last X days, Specific day	Shown for Custom widget settings only, and only when the widget's data list has a field detected or manually marked as Date. In inherited mode the source Home list owns the period, so it is displayed as inherited instead of duplicated. The date constraint is applied before aggregation/read limits. Last X days uses a directly editable number; Specific day uses the calendar. A field whose name looks like a date but is still typed as Text does not activate period filtering.
Refresh	Loading mode	Advanced picker: Cache or Live query	Basic uses the safe cached default. Advanced can explicitly select Live query.

Section	Control	Type	Purpose
Refresh	Auto refresh value / Every X min	Toggle + stepper 1 to 1440	Refreshes the cached widget value through the same app-level scheduler and serialized queue used by lists. Save performs an immediate refresh and shows any source/filter error. The resulting snapshot is revisioned so iPhone and Watch reject an older value that arrives after a newer one. The interval is a target/minimum due time: iOS decides when a suspended app receives background time. Cifru has no required developer relay, so it does not promise exact unattended minute-by-minute delivery.
Conditions	Field, Condition, Value	Guided pickers, date picker, or numeric field	With Custom widget settings, Basic infers field type from discovered schema. For a Date field, choose a fixed date with Equals/Greater or equal/Less or equal, or a dynamic period: Today, Yesterday, This week, This month, This year, or Last X days. With an inherited list, this section becomes Additional conditions and narrows the inherited result without duplicating its date period. Dynamic periods are recalculated at every refresh and applied before database aggregation or connector row limits. Advanced can also edit the raw key and field type.
Optional alarm	Enable alarm, value type, operator, comparison value	Toggle + guided pickers/field	Evaluates the newly refreshed widget value as a number or text. Numeric operators are Equals, Greater than, Less than, Greater or equal, and Less or equal. Text operators are Equals and Contains.

Section	Control	Type	Purpose
Optional alarm	Notification behavior	Picker: Brief notification / Alarm in Cifru + notifications	Brief creates one local notification. Alarm in Cifru opens a full-screen acknowledgement view and repeats its local sound until Stop or Snooze while Cifru is visibly active. If the app is in the background or the device is locked, Cifru uses a finite standard notification sequence with Stop/OK and Snooze instead of background audio.
Optional alarm	Sound, snooze, cooldown, hide value on Lock Screen	Pickers/toggle	The foreground in-app sound uses the selected standard/silent behavior. Background delivery remains subject to Silent Mode, Focus, notification permission, and iOS scheduling. Edge triggering and cooldown prevent repeated alerts while a condition remains true. Hiding the value protects it on Lock Screen notifications and accessory widgets.
Optional alarm	Test alarm	Button	Starts the full-screen in-app alarm when acknowledgement mode is selected and Cifru is active, and also prepares the finite notification fallback when permission is available. It does not read or change a source.
Sort	Field, Direction	Guided pickers; technical key/type in Advanced	Shown for Top list and in Advanced. Sorting affects ranked rows; it does not change a scalar total.
Toolbar	Save	Button	Saves and immediately refreshes the widget.

Widget alarm limits: Cifru evaluates an alarm only after a fresh source read completes successfully; stale cache values and failed refreshes do not create a new alert. Continuous sound is an in-app foreground experience only. When Cifru is not active, delivery uses finite best-effort iOS local notifications and exact timing/sound are controlled by the operating system. Cifru does not run background audio, request Critical Alerts, bypass Silent Mode or Focus, operate an alarm server, or claim medical, emergency, safety, or life-critical suitability.

Opening Filters, Conditions, And Sort Rules

For supported SQL and connector reads, fixed filters and a selected opening-filter value are passed to the source before the final row cap, so Cifru does not need to download every warehouse/store just to show one selection. Opening-filter choices use bounded distinct-value discovery where supported. With Cache mode, the selected value is also applied locally to the full cached data and the resulting page partition is retained without replacing the full cache. Free-text search operates over all rows already loaded for that list, before UI pagination. If a list is limited to 100 rows, search can inspect only those fetched rows; increase Max rows, turn the limit Off, or use a source-side period/filter when broader coverage is required.

Concept	Available choices	Best practice
Field type	Text, Number, Date	Use Date for invoice dates/deadlines, Number for amounts/quantities, Text for names/status/codes.
Filter condition	Equals, Contains, Greater or equal, Less or equal	Contains is useful for search text. Greater/less is useful for dates and numbers.
Sort direction	Ascending, Descending	Descending for latest invoices/orders, top values, newest dates. Ascending for alphabetic lists or oldest first.
Opening filter Allow All	On/Off	Keep On when the user should be able to open all rows; turn Off when choosing a value is mandatory.

Browsing Data And Export

When a Home button opens a list, the user sees rounded mobile cards with title/subtitle/badge fields, an ordered grid of selected values, opening filters, search, sorting, row buttons, and paging. When **Require filter selection when opening** is enabled, Cifru waits until every required filter has a value or the user chooses Show all. When it is disabled, the list opens immediately and those controls remain optional filters above the rows. Opening a row creates one composite workspace: hero header, important values, grouped metadata, and related sections. The first automatic Details link is inline and later links are collapsible; Studio can instead choose tabs or a separate screen. Child rows open in an inspector without losing the parent context. The same layout engine is used for a list opened by a Details or Open-list sub-button, and Studio Actions provides a direct Customize-cards link for that destination screen.

Action	What happens
Refresh this list	Pulls fresh rows according to the linked list/source configuration.
Export Excel-compatible XLS	Creates a spreadsheet-friendly export using current search, filters, and sort.
Export PDF	Creates a readable PDF export for quick sharing. Large PDF exports are limited for mobile performance.
Share sheet	The user chooses Mail, Messages, AirDrop, Files, or another installed app. Cifru does not upload exports to a Cifru server.
Pagination	Rows per page is controlled in Home button editor. This avoids huge mobile lists.

Settings

Lists are managed only from Build, where their screen, relation, and field context is visible. Settings intentionally does not duplicate the list inventory. Cache & Sync still shows per-list cache/refresh diagnostics because those controls describe runtime state rather than list configuration.

Section	Control	Type	Purpose
Subscription	Manage subscription	Navigation link	Opens Free/Starter/Pro screen with purchase and restore.
Language	App language	Picker: Automatic, Romanian, English, Spanish, French, German, Italian, Portuguese, Polish, Hindi, Simplified Chinese, Urdu, Nepali, Bengali, Indonesian, Arabic, Vietnamese	Automatic follows the phone language when it is supported and otherwise falls back to English. Core navigation, Settings, source setup, Build, widgets, cache/security, and Help are localized; technical product/standard names stay unchanged. Missing secondary-language copy falls back to English rather than mixing languages.

Section	Control	Type	Purpose
Appearance	Theme	Picker: Dark, Light, Eye Comfort	Eye Comfort is the default for new installations. Existing saved choices remain unchanged. Older System settings are treated as Dark.
Date format	Date display	Picker: Automatic, dd-MM-yyyy HH:mm, yyyy-MM-dd HH:mm, Date only	Automatic follows the phone's current region, time zone, short-date, and short-time settings across cards, details, previews, widget rows, and refresh timestamps.
Data sources	Source timeout (seconds)	Numeric text field, 5 to 600 seconds, default 180	Maximum wait time for source testing, Browse tables/endpoints, preview, and refresh. Increase for slow VPN/database/API connections; lower it when a bad source should fail faster.
Safety and transparency	Query transparency	Navigation link	Shows the effective read-only request plan for every Home screen and Details screen. SQL entries display the generated SELECT statement with fixed filters, sort, and limit. Every Details sub-button and card enrichment also shows its parameterized child read with a selected-card placeholder instead of a real row value. Other connectors show the method, public path, cache mode, filters, relations, and enrichments. Credentials, tokens, passwords, private hosts, and sensitive query parameters are masked. The plan can be copied for review but cannot be edited or executed from this screen.
Status	Refresh all data	Button	Refreshes all lists and widgets.
AI Configurator	Enable AI assistant	Toggle	Enables/disables the AI dashboard draft assistant.

Section	Control	Type	Purpose
AI Configurator	AI Configurator settings	Navigation link	Opens provider selection, API key storage controls, current model picker/base URL for external providers, and External AI Documentation Research settings.
AI Configurator settings	Provider	Picker: Apple On-Device Recommended, Mock / Local Templates, OpenAI, Claude, Gemini	Apple On-Device is preferred when available and requires no API key. Mock uses local templates. OpenAI, Claude, and Gemini are optional external providers.
AI Configurator settings	OpenAI / Claude / Gemini API key	Secure field + Save key + Clear key	Stores the user-provided key in Keychain. A visible saved-key status confirms that the key will be reused automatically. Changing the model for the same provider does not clear or require re-entering the key. Only Clear key deletes it.
AI Configurator settings	Available model	Provider-backed picker + Refresh model list	When Settings opens, the provider changes, or the key is saved, Cifru reads the current model catalog directly from OpenAI, Claude, or Gemini. If a saved model was retired, the first available model is selected. If loading fails, the saved model remains as a fallback and the existing Keychain key remains active.
AI Configurator settings	Base URL	Text field	Editable for compatible provider endpoints. The model catalog is read from this configured provider base URL.

Section	Control	Type	Purpose
External AI Documentation Research	Enable external research	Toggle	Disabled by default and available only with OpenAI, Claude, or Gemini. Requires user approval before sending the application/system name, public documentation URL, or public documentation text. Research searches official producer/vendor manuals first and reports the documents and assumptions used.
Help / AI Configurator	Ask an external AI	Navigation link	Separate from the built-in AI Configurator. Without a configured external key, it opens the localized copyable prompt. With a stored OpenAI, Claude, or Gemini key, it opens a persistent explanatory chat using that same provider, model, base URL, and Keychain credential.
External AI chat	Provider / New conversation	Menu	Switches among external providers whose keys are already stored, or deletes the protected local transcript and starts a new conversation.
External AI chat	Message + Send	Multiline field + icon button	Sends a bounded recent conversation window, safe metadata, and locally retrieved relevant sections of the complete public Cifru guide after first-use consent. The assistant may search public vendor manuals, explains setup, and cannot apply a configuration.
Siri	Siri command hints	Text examples	Shows examples such as "Help me configure Cifru", "Open Cifru sources", and "Ask an external AI with Cifru".

Section	Control	Type	Purpose
Cache & Sync	Cache summary and synchronization diagnostics	Navigation link	Shows cached row count, cache size, stale lists/widgets, refresh actions, clear cache, and the current refresh queue state: running, queued, last successful update, last failure, and retry status. Automatic list and widget refreshes share one serial, deduplicated queue so several due caches do not open competing source reads at once. A user-triggered refresh is placed ahead of queued background work, short cooperative pauses leave room for navigation and taps, and failed automatic reads use progressive retry delays. Opening Home only schedules work that is actually due; it does not wait for that work and automatic jobs do not show the global interactive busy indicator. Cache writes are coalesced and performed with utility priority. Cifru retains an already pending iOS background request when it can run sooner instead of cancelling and postponing it on every lifecycle event. A short background opportunity processes at most two fairly selected reads and prioritizes an external widget within widget work. The screen audits the complete path separately: iOS scheduling, source/list read, widget calculation, App Group snapshot revision, WidgetKit timeline reload, and Watch delivery/confirmation. Advanced mode also exposes an optional External relay + iOS fallback architecture. The app-side relay parser accepts only internal source IDs, a monotonic revision, an event type, and an optional timestamp; it rejects rows, credentials, tokens, private URLs, and business data. Production relay activation separately requires APNs capability/provisioning and a managed relay deployment. The normal iOS-managed path

Section	Control	Type	Purpose
			always remains the fallback. Background refresh remains best-effort and runs only when iOS grants execution time; configured minute intervals are due thresholds, not guaranteed timers.
Home Screen, Apple Watch & CarPlay widgets	Refresh snapshot widget every X min	Stepper	Requested WidgetKit timeline interval for the local external snapshot. iOS/watchOS decide actual background execution time, so the displayed timestamp and stale state always refer to the cache value being shown. Generic feeds retain complete bounded snapshots for correctness; SQL and other capable connectors push supported filters, date periods, sorting, aggregation, and limits to the source before materializing rows.
Home Screen, Apple Watch & CarPlay widgets	Hide values in external widgets	Toggle	Protects sensitive values on lock/watch/car displays.
Home Screen, Apple Watch & CarPlay widgets	Update widget snapshot now	Button	Publishes the latest selected widget values atomically to the shared local snapshot and reloads every registered iPhone and Watch widget kind. On iOS 17+/watchOS 10+, edit the external widget and choose any configured Cifru widget by its own name; older iOS versions use the compatible static snapshot widget.
Local data	Clear cached rows	Button	Deletes cached rows and cached widget values, not the configuration.
Local data	Reset workspace and secrets	Destructive button	Deletes configured sources, cached data, and stored secrets from this device.
App mode	Branding and app mode	Navigation link	Set company name/logo and enable simplified app mode.

Section	Control	Type	Purpose
App mode	Preview app mode	Navigation link	Preview the simplified configured app experience.
Privacy & Security	Advanced mode	Toggle	Basic keeps guided selectors and common calculations. Advanced additionally shows internal field names/types, manual operators, icon names, token headers, row paths, search keys, advanced widget kinds, cross-source matching controls, and explicit Cache/Live selection.
Privacy & Security	Require Face ID / passcode	Toggle	Requires local device authentication before entering the app.
Privacy & Security	Show security warnings	Toggle	Shows/hides configuration warnings.
Privacy & Security	Security review	Navigation link	Shows source/list/security findings.
Legal	Help, Legal, Email support	Links	Support and legal documents.

App Mode / Use As App

Path: `Settings > Branding and app mode`. This is how a configured workspace can feel like the user's own native app.

Control	Type	Purpose
Company name	Text field	Name shown in the app mode header.
Company logo	Image picker	Logo shown in app mode and on the locked Face ID/passcode screen when configured.
Remove logo	Destructive button	Clears the custom logo.
Use as app	Button	Enables app mode and hides builder tabs.

Control	Type	Purpose
Exit app mode	Button in app mode settings	Returns to the builder.

Subscription Limits

Plan	Limits	Notes
Free	1 data source, 2 tables/lists from that source, basic widgets, local cache, export, 0 data links.	Free users can still build useful buttons and widgets for one source, but they cannot create unlimited tables/lists.
Starter	3 data sources, unlimited tables/lists, and 3 data links.	Good for small dashboards that need several sources and a few details links.
Pro	Unlimited data sources, unlimited tables/lists, unlimited data links, advanced widgets and dashboards.	Pro and old Business entitlement names resolve to Pro.

StoreKit 2 loads the Starter and Pro product names, descriptions, localized prices, and billing periods from App Store Connect. Cifru does not invent fallback prices. If products are unavailable, the screen shows an error and Retry. Restore Purchases uses the user's Apple Account, and Privacy Policy plus Terms of Use remain available on the same screen. Local Debug builds can use a Pro development entitlement; Release and TestFlight builds require an active verified StoreKit entitlement. After a successful verified purchase or transaction update, Cifru activates the verified plan immediately and then reconciles it with `Transaction.currentEntitlements`, so temporary App Store sandbox propagation delay does not keep the previous plan on screen. TestFlight purchases run in Apple's sandbox and do not charge real money.

If a subscription expires, is revoked, or moves to a lower plan, Cifru returns to the applicable lower plan and keeps the user's local configuration so it can be restored later. Sources, lists, relations, pages, and widgets above the active plan limits are visibly locked and cannot test a connection, browse tables/endpoints, preview or read rows, refresh in the foreground or background, appear as usable Home content, publish external widget data, or be sent to the AI Configurator as usable metadata. Reactivating an eligible entitlement restores access to the preserved configuration.

Settings path: `Settings > Subscription Plans`. Product IDs: `com.madalindumitru.cifru.starter.monthly` and `com.madalindumitru.cifru.pro.monthly`.

Siri, App Shortcuts, And Apple Intelligence Readiness

Cifru exposes a small set of safe App Intents/App Shortcuts so Siri and system surfaces can help users reach the right setup area. This is the public Apple-supported path for Siri/Apple Intelligence integration on current iOS releases and is the safest future-compatible anchor for newer iOS versions.

Shortcut / Intent	What it opens	Safety boundary
Help me configure Cifru / Open Cifru start guide	Start guide sheet with the recommended setup path and templates.	Does not read credentials, cached rows, or source data.
Open Cifru sources	Sources tab, so the user can add, save, and test a source manually.	Does not create a source automatically and does not test without the user opening/saving settings.
Open Cifru build / Build my dashboard in Cifru	Build tab, where lists, Home buttons, Details, links, and widgets are configured.	Does not run source queries by itself.
Ask AI with Cifru	Ask AI screen with the copyable prompt and documentation link.	Only shows a local prompt. The user chooses if and where to paste it.

Important: Siri/App Shortcuts are navigation and guidance helpers. Cifru does not send database credentials, API keys, cached rows, exports, or imported local files to Siri or to a Cifru backend. The app does not depend on the legacy SiriKit capability for this flow; it uses App Intents/App Shortcuts.

Widgets, Apple Watch, And CarPlay

Cifru external widgets read a local snapshot created by the iPhone app. They do not open network connections directly. This is important for safety, App Store review, and privacy.

- iPhone Home Screen widgets can show selected dashboard widgets.
- Lock Screen accessory widgets can show compact values.
- Apple Watch uses WatchConnectivity to receive every widget marked for external display, then Watch complications read the Watch-side snapshot. Cifru publishes the newest revision through application context, sends it immediately while the paired Watch is reachable, and queues the latest revision for background delivery without cancelling a transfer already in flight. It uses the dedicated current-complication transfer when watchOS reports available

budget and a background user-info transfer otherwise. A failed latest background transfer receives two bounded retries with backoff instead of being silently discarded. The Watch extension retains a WatchConnectivity background task until pending content has been processed, accepts only a newer revision, saves it atomically, reloads its timelines, and returns the applied revision to iPhone. If a newer iPhone snapshot appears while one transfer is pending, only that newest revision is sent next. Settings distinguishes a snapshot published by iPhone from a revision actually confirmed by Watch. Delivery remains best-effort and scheduled by iOS/watchOS rather than guaranteed at an exact minute. On watchOS 10 or later, the gallery exposes one preconfigured Cifru recommendation for every selected widget, named with the title configured in Cifru; the title does not depend on the Watch language.

- To add a Watch complication: enable external display in the Cifru widget editor, update the widget snapshot, open Cifru once on the Watch to allow synchronization, long-press the Watch face, tap Edit, choose a complication slot, and select the desired Cifru entry by its configured title. watchOS 9 shows the first selected widget as a compatibility fallback.
- Inline and rectangular Watch layouts preserve the configured title and scale the value to fit. Circular and corner layouts use a compact value such as 57.2K while keeping the full title/value available to accessibility.
- Full widget values follow the device locale. Thousands grouping is preserved, integers show no decimal part, and non-integers show exactly two decimal places. For example, a Romanian device displays `57154.22` as `57.154,22`, while `57154` displays as `57.154`.
- CarPlay support is widget-only through WidgetKit small widgets where supported by iOS. Cifru is not a separate CarPlay app target.
- Default external widget snapshot refresh request is 5 minutes, but iOS controls the exact timing.

Cache Guidance

The configured refresh interval means that data becomes due for refresh after that time; it is not an exact background timer. iOS decides when Cifru may run. The default iOS-managed mode catches up at every granted background opportunity and whenever the app becomes active. An optional metadata-only relay mode can wake the same bounded read-only queue after a source-change event, while retaining the iOS-managed fallback. The relay event never contains source rows or secrets. Enabling a real production relay is a separate deployment task because it requires APNs provisioning and a managed service.

Even when **Limit number of rows** is Off, Cifru applies adaptive device protection. An unbounded or unusually high-limit read receives a 15-second safety window and an internal ceiling of 50,000 rows. A small, fast, or source-filtered result can still complete without an explicit row limit. A read

that exceeds the safety window or reaches the ceiling is stopped before rows are cached or rendered and shows an actionable message. Select a date field and period, add a filter, or enable a smaller explicit row limit before retrying.

Source/list type	Recommended mode	Reason
Reliable SQL over LAN/VPN, small row count	Live query or Cache	Live is convenient when the connection is stable. Cache reduces repeated mobile database access.
Large SQL tables/reports	Cache with Max rows and refresh interval	Protects the phone and source database from huge repeated queries.
FTP/feed files	Cache	Feed files are often static for a period and should not be downloaded constantly.
Ecommerce/API sources	Cache with reasonable Max rows	APIs often have rate limits and paging.
Widgets / Watch / CarPlay snapshot	Cache or local snapshot	External widgets should use local data only.

Troubleshooting

Problem	Likely cause	What to tell the user
Secure connection could not be trusted	Private/self-signed certificate, mismatched host name, missing public chain.	If it is their own SQL server, enable Trust server certificate, or install a valid public certificate. For new SQL Server sources the toggle defaults on; if it was turned off, Cifru shows a hint after the failed test. Do not enable it for unknown servers.
SQL port appears wrong	Port field should be plain numeric text.	Type the full port directly, for example 1433 or 8433. Do not use separators such as 1.433.
No rows found	Wrong table/path/sheet, authentication, rows path, or source returned an empty response.	Use Check connection, Browse tables/endpoints/sheets, search/select the object if available, tap Save selection & preview rows, verify endpoint/table/sheet, and set Rows path if JSON/OData/GraphQL response is nested.

Problem	Likely cause	What to tell the user
Browse tables/endpoints or preview times out	Slow VPN, source server, large metadata query, slow API, or unreachable host.	Open Settings > Data sources and adjust Source timeout. Default is 180 seconds; valid range is 5 to 600 seconds.
Connected, but not OK	Login/basic connectivity succeeded, but catalog, columns, endpoint rows, sheet, or feed structure could not be verified.	Open the source and read the exact second-stage error. Do not build or ask AI to guess until Save/Check connection reaches OK.
AI response is incomplete / JSON is cut	The provider exhausted its output budget or returned a malformed response before closing the JSON object.	Cifru discards the partial response and retries once with a compact request without echoing the cut JSON. If the retry also fails, choose another current model/provider or try again; no draft is created or applied.
AI found documentation but not the requested object	The official manual describes the requested flow, but the configured source did not expose the matching concrete table/view/endpoint or its detail object.	Check read-only schema permissions and source setup. Cifru intentionally stops instead of substituting an unrelated document type.
NIOCore.ChannelError / connection closed during MariaDB or MySQL discovery	Transport channel closed between connection and schema discovery because of a transient network/VPN event, TLS negotiation, proxy/firewall behavior, or server connection limits.	Cifru retries once with a fresh connection and uses three compatible table/column discovery methods. If it still fails, verify Require/Prefer TLS, certificate settings, VPN/firewall and server connection limits. A working login alone does not prove catalog/preview access.

Problem	Likely cause	What to tell the user
XML feed shows no fields	The repeated record element is not named `item`, for example `Produs`, or the XML is malformed.	Leave the row name on its default and retry; Cifru now detects a repeated structured element automatically. For ambiguous XML, enter the exact row element manually.
REST JSON returns one wrapper instead of rows	The row array is inside a response envelope.	Cifru recognizes common data/results/items/records/rows/value/nodes envelopes. For a custom structure, set Rows path explicitly.
Details button or cross-source completion shows no matches	The two explicitly selected fields do not contain the same normalized value format.	Open the Details or cross-source linking wizard, choose the second source/list, select one concrete field on each side, and check the bounded representative match rate. A field's optional semantic meaning can suggest candidates, but it does not create the link.
Filter does not find expected data	Rows were not fetched due to Max rows/cache limit, wrong field type, or wrong field key.	Increase Max rows, refresh the list, use the field picker, and set Type to Date/Number when appropriate.
Widget value looks old	The source read may be overdue, the external snapshot may still be pending, or Watch may not yet have applied the latest revision.	Open Settings > Cache & Sync and compare the latest iOS execution with Apple Watch sync. Use Refresh widget cache now, then Update widget snapshot now. Cifru keeps the newest revision queued and the Watch extension processes pending WatchConnectivity content in background when watchOS grants time. A pending Watch confirmation still means iOS/watchOS has not completed best-effort delivery yet; opening Cifru once on Watch requests the latest application context and an immediate reachable-session copy.
Source row says Configure	User added source but did not save/test successfully yet, or the latest automatic test failed.	Open the source, complete required fields, then tap Save or Check connection. Save keeps settings and runs an automatic test.

Step-By-Step Examples

Example 1: SQL Server Customer Invoices With Details

1. Go to `Sources > Add sources > Databases > SQL Server`.
2. Open the new source. Fill Host, Port, Database / service, Username, Password, TLS. Use a read-only SQL user.
3. For new SQL Server sources, Trust server certificate is on by default because many private company servers use internal certificates. Keep it on only when the server is controlled by the user; otherwise prefer a valid public certificate.
4. Tap Save or Check connection. Both save the current settings and run a connection test. Wait for OK.
5. Go to `Build > Home screens > Choose source and table/path` and choose the tested SQL source.
6. Tap Browse tables. A modal opens over the assistant. Use the search field or table list to choose the invoice table/view. The selected SQL object fills Table/view to read directly and is displayed once as schema.table, for example `dbo.accesex_facturi_clienti`.
7. Tap Save selection & preview rows.
8. In Choose fields, set friendly labels such as Invoice number, Customer, Date, Total, Store. Set Type Date for invoice date and Number for totals.
9. Optionally assign semantic meanings such as Invoice ID, Customer ID, Store/Location, or Date. These labels help formatting and later suggestions; they do not create a relationship by themselves.
10. In Add to Home, name the button Customer invoices and tap Add button to Home.
11. In the Customer invoices Home screen, keep Sort set to the invoice date field descending and set Max rows to a reasonable value such as 100, 500, or 1000. For generated SQL reads, Cifru applies the date sort before the row limit, so the app fetches the latest invoices instead of an arbitrary set.
12. Create or configure a second list for invoice lines. Preview it and map Invoice ID, Product ID, EAN/Barcode, Quantity, Price.
13. Return to the Customer invoices Home screen. Tap Add guided Details or Details / row action > Guided Details.
14. Choose invoice lines as Details list/table. Match parent `invoice_id` to child `invoice_id`. Use Suggest fields automatically and check matching rate.
15. Save. Opening an invoice row should now show a Details button with matching invoice lines.

Example 2: Product Enrichment From WooCommerce

1. Add the WooCommerce source with Store URL, Consumer key, and Consumer secret.
2. Check connection.
3. Go to `Build > Home screens > Choose source and table/path` and create a Products button from known store data or an API path.
4. Map product ID, SKU, EAN/Barcode if available, product name, image URL, price, stock status.
5. On an invoice-lines list, optionally identify EAN/Barcode or SKU by its semantic meaning so Cifru can suggest a likely candidate.
6. For one product match per invoice line, use `Build > Home screens / buttons > select invoice lines screen > Fields from other lists`. Choose the enriching source/list, the concrete EAN/SKU field in the current screen, and the equivalent concrete field in the selected source. Review the match rate before adding product name, image URL, price, or stock status to the same card.
7. If the product should open a separate product screen, use a row button named Product with Show matching details.

Example 3: FTP CSV Feed Dashboard

1. Go to `Sources > Add sources > Data feeds > FTP file`.
2. Fill FTP host, Port, Username, Password, Remote file path, and File format CSV or AUTO.
3. Check connection.
4. Go to `Build > Home screens > Choose source and table/path`, choose/preview the file, map fields, and create a Home button. For XLSX files, tap Browse sheets and select the sheet that should become the list.
5. Set list Loading to Cache, Auto refresh cache On, Every 15 or 60 minutes depending on feed update frequency.
6. Add widgets such as Count, Sum, or Top list using filters and sort rules.

AI Configurator

Path: open `Build`, then tap the visible `AI Configurator` text action in the upper-right toolbar. This screen lets a user describe a dashboard in normal language. If the assistant is disabled, or an external provider is selected without a saved API key, this toolbar action opens `AI Configurator settings` directly so the user can enable and complete setup first. The selected provider returns only declarative JSON. Generate draft automatically discovers concrete schema, compiles mandatory main/Details/lookup topology locally, validates the JSON, runs bounded read-only functional tests through the real connectors, performs at most one safe functional repair-and-

retest pass when needed, and then shows the final JSON/dashboard preview. The user still decides whether to tap `Apply configuration`. While generation is active, the red Stop control also displays a monospaced elapsed-time counter; it is elapsed time, not a simulated completion percentage.

Community and configuration library

Path: `Build → Community and configuration library`. `Library` contains only public-catalog actions: explore approved packages or select one or more of the user's Home-button trees and submit them for moderation. `Direct transfer and backup` is separate: it imports or exports a `.cifruconfig` attachment through Mail, AirDrop, or Files without publishing anything. A shared tree includes its required Details/sub-buttons, lists, relations, enrichments, related widgets, and used common fields. Application/system name, configuration language, and country coverage are mandatory. Multiple countries are supported; `ALL` is available only for configurations that include English.

Every import makes each Home tree an explicit selectable checkbox, reports how many are selected, and offers Select all / Deselect all. The full downloaded JSON and the exact selected subset are shown separately. The user maps symbolic source slots to already configured local sources, then Cifru confirms direct-database object fields before bounded reads through the real connector. Compatibility failures identify the affected list/object and missing fields instead of exposing a generic driver error. Apply remains visibly locked with a reason until selection, plan, mapping, and read-only tests pass. Custom-query components cannot be imported in Free or Starter regardless of whether the file came from Community, Mail, AirDrop, or Files. Packages never carry source credentials, private endpoints, cached rows, or subscription rights. The last successful import can be undone from the Community screen.

The optional Community account is needed only for publishing, reviews, and contribution management. Cifru reads the public service health status and displays only social providers actually configured by the Community operator. Mobile authorization uses a browser session, state and PKCE; the resulting Cifru Community token is stored in Keychain. Browsing, package download, direct import, and private export remain available when no login provider is active. Publishing can include up to eight optional screenshots selected through the system photo picker. Cifru normalizes them to JPEG on device; the service validates dimensions and format, rewrites them without embedded metadata, creates thumbnails, and publishes them only after moderation. Configuration languages and countries use compact searchable multi-select screens; country code `ALL` remains mutually exclusive with individual countries.

The normal flow is designed for non-programmers. Tap one of the request examples or write a short request, leave source choice on *Automatically choose relevant sources*, then generate. During generation, the same action becomes *Stop generation* and cancels the active provider/check

pipeline without deleting the previous draft. Cifru shows only the real current operational phase, not model chain-of-thought. Regenerate keeps one previous draft that can be restored. The result is human-first: visual Home cards, nested Details and widgets appear before an exact change summary; technical topology and raw JSON are collapsed until opened. Apply has a separate confirmation and a successful apply can be reversed once with *Undo last AI apply*.

Apple On-Device has a smaller practical context budget than external providers. When necessary, Cifru sends it a relevance-ranked subset that keeps the strongest live main/detail candidates and relation/enrichment fields for every independently requested flow. The UI shows both full prepared counts and focused counts. Local graph completion and validation always use the full discovered schema. External BYO-key providers normally receive one request, but malformed output or one repair after a real functional test may require an additional provider request and may incur provider charges.

For best results, the user should write which application, database, service, or system they want to integrate, which Cifru data source/list should be used if the AI cannot infer it, and which Home buttons, row Details buttons, filters, widgets, or exports they want. If External AI Documentation Research is enabled and approved, the external provider searches official producer/vendor manuals first. Cifru also supplies generic implementation notes describing how its installed SQL, feed/file, REST/GraphQL/OData, ecommerce, Details, filtering, limiting, and enrichment components actually behave. The AI must cross-check every proposed object and field against exact safe metadata and list manual titles/URLs and uncertainty under Documentation used and assumptions. The initial request contains no source rows. An optional automatic-repair request may include only the bounded identifier sample described in the privacy section; it never includes credentials, names/contact data, invoice contents, descriptions, prices, totals, stock, quantities, or private URLs.

If the user has added and fully verified a source but has not manually created a list/table, AI Configurator can still start. Cifru discovers schema locally: SQL table/view names and column names/types through fixed read-only catalog queries, and feed/API/XML field or tag structure through a bounded local preview. MySQL/MariaDB discovery tries `SHOW TABLES`, `SHOW FULL TABLES`, and `information_schema.tables`; columns use `SHOW COLUMNS`, `DESCRIBE`, then `information_schema.columns`. A transient NIO channel closure is retried once with a fresh connection. HTTP/FTP transport interruptions are also retried once when safe. Object names are ranked separately for each requested flow. Cifru contains no executable profiles keyed by ERP, ecommerce product, application, table, endpoint, installation prefix, or field name. Public manuals can improve the external provider's interpretation, but every candidate still comes from the live source. A connection is only a profile, not a readable list: Cifru never sends generic `_source_setup` metadata to AI as a usable list object. It resolves the selected connection to a concrete table, view, endpoint, sheet, feed, file, or vendor object and requires the returned list to reference that object. If the concrete structure cannot be read, generation stops before calling AI

and distinguishes transport interruption, permission denial, authentication, TLS, invalid response, and missing-object cases where possible. Cifru also blocks an AI-created SQL list whose table/view target is empty. If the user selects one primary source, Cifru can include verified auxiliary lookup sources such as feeds, files, object storage, REST/GraphQL/OData APIs, and ecommerce connectors. Initial generation sends schema metadata only. Full row/feed contents, URLs, hosts, credentials, tokens, connection strings, invoices, customers, prices, stock, and quantities are never sent. One later automatic repair may include only the optional bounded identifier sample described below.

After an AI provider returns JSON, Cifru sanitizes it and tries to repair common draft mistakes before validation. It first reconciles each list with a concrete discovered object using the list's requested role (main, Details, lookup/enrichment), dashboard/list/relation titles, configured source alias and root connection, exact field coverage, and the provider's declared object references. This prevents a semantically different but readable table from replacing the requested business object merely because both expose fields such as `id_document`. Import, staging, queue, buffer, and write-back objects are strongly rejected as automatic substitutes unless the user explicitly asks to inspect that exact read-only surface. If top-level `dataSources` and list `dataSourceId` values contradict each other, the repaired graph is made internally consistent before testing. The generic provider repair runs before Cifru's deterministic request completion, so it cannot later reverse a locally verified Details or enrichment direction. When the user explicitly selected a concrete table, endpoint, sheet, feed object, or configured list, that exact object remains the visible primary list before Cifru adds Details or cross-source fields; a documented integration profile can still replace it when the prompt explicitly requests that documented business object. Examples of other repairs: if the AI invents generic fields such as `title`, `status`, `amount`, or `valoare`, Cifru maps them to the closest fields in the discovered schema or removes them if no safe match exists. If a widget points to a list created in the same draft, Cifru accepts that list reference. If an enrichment is backwards, such as an XML feed trying to enrich invoice lines, Cifru can flip it so the visible invoice-line card is enriched from the feed. If an AI provider mistakenly uses a raw source ID as `parentListId` or `childListId` in a relation/enrichment, Cifru creates an intermediate lookup list for that source and rewrites the relation/enrichment to the generated list ID. Feed/API/XML display and enrichment fields must exist in the concrete detected schema. Guessed fields such as `LinkEmag`, `competitorUrl`, `imageUrl`, `EAN`, or `SKU` are removed when that exact source did not expose them.

Provider completion is checked before any JSON fragment is accepted. If OpenAI, Claude, or Gemini reports an incomplete/max-token response, Cifru ignores the partial text and makes at most one compact retry. The retry uses the original request and the same safe metadata; it does not echo the malformed partial JSON. The same retry is used when a provider claims completion but returns an unclosed or non-object JSON document. If the second response is still malformed,

Cifru discards both responses, shows a clear error, and leaves Apply disabled. Truncated JSON is never sent into the normal mapping-repair pass.

A complete JSON response that merges independent requests, chooses the wrong live object, omits a Details edge, reverses a relation, or omits a requested enrichment does not consume a second provider request. Before object binding, Cifru turns numbered and natural-language clauses into a vendor-neutral intent graph. Each business flow receives its own Home role; inherited phrases such as “la fel” keep the requested Details shape without copying the previous business object. Cifru then deterministically rebuilds the graph from concrete objects and compatible semantic field aliases already discovered on the device. The reconstructed graph is validated for both technical readability and requested-flow coverage and must still pass the production-connector dry-run.

Finding readable tables is not enough: every requested flow must have an independently matched Home screen, every requested Details action must use a compatible child object and real relation fields, and every requested opening filter must exist on that flow. A global dashboard title is never used to bind every list, and a relation for one parent/child pair cannot overwrite another sub-button. If the concrete graph cannot be built from locally discovered safe metadata, generation stops and explains the missing flow instead of silently returning fewer buttons.

Enrichment lookups are chosen conservatively and are transport-neutral. A per-item lookup may come from any configured read-only source, including SQL Server, MySQL/MariaDB, PostgreSQL, Oracle, an ecommerce connector, API, feed, FTP/object storage, or local file, when its concrete item identity matches the detail rows. Cifru prefers exact or semantically compatible pairs such as `cod_produs` to `EAN`, `barcode` to `ean_code`, or matching SKU/product IDs. A wide business table joined only through a header/document identifier is a master-detail relation, not a lookup, and is not attached as card enrichment. When the user asked for extra card fields but no configured source exposes a compatible lookup schema, Cifru still generates the main screen and Details, explains the missing lookup in a warning, and lets the user connect or map the lookup source later, instead of failing the whole generation.

For every flow, Cifru removes a conflicting provider-proposed enrichment between the selected detail and lookup lists, then creates one locally validated edge in the visible-detail-to-lookup direction. This prevents a reversed or duplicated AI edge from leaving a valid field name attached to the wrong source. Verification is tied to exact source identities, so two configured instances of the same application cannot invalidate or redirect each other's dashboard.

Required topology is not delegated entirely to the AI provider. After parsing the provider's declarative JSON, Cifru interprets common non-technical requests such as "when I open one, show what it contains", "put the products inside each order", "show the conversation for each ticket", or "match the picture from the other feed by product code". It then completes a missing main-to-Details relation or cross-source enrichment locally, using only exact discovered fields,

compatible semantic roles, and built-in Cifru components. This works the same way for databases, ecommerce APIs, REST/GraphQL/OData, feeds, FTP/object storage, and local files. It can match vendor-neutral identifiers such as `ticket_id`, `project_id`, `budget_id`, `PO_ID`, UUIDs, documented ID pairs, and EAN/SKU families. A plain ambiguous `id` is not accepted as a fallback unless the local metadata gives it a compatible semantic role.

The original user request remains the semantic authority throughout automatic repair. Cifru sends a separate technical repair prompt to the provider, but it does not rebuild intent from that repair text, a global dashboard title, or a previously wrong draft. Requested main/detail roles and their Details edges are reconstructed locally after every provider response. The final check compares compatible fields actually discovered in the installation; safe aliases may match when local metadata assigns the same semantic identity. Provider output cannot remove, merge, or redirect a requested flow.

Cifru also normalizes requested scope before preview. A provider-created relation is removed when the user did not ask for Details; an enrichment is removed when the user did not ask to combine sources; and unrequested or duplicate widget types are removed. Short intent words are matched as complete words, so a request containing `categorie` is not misread as the Romanian count word `cate`. Common Romanian singular/plural forms such as `imagine`, `imagini`, and `imaginile` are recognized, while explicit refusals such as "nu vreau indicator" take priority over positive keywords elsewhere in the sentence. Phrases such as "show the total field on the card" do not create a metric widget, and "show the existing link column" does not create a cross-source mapping. The resulting graph is validated again and then sent through the local read-only dry-run. If the user's intent is still ambiguous or no compatible concrete objects/fields exist, Cifru stops with a focused setup message instead of guessing.

Cifru then runs an automatic local dry-run through the same production connectors used by the generated screens. It tests every list/widget with small row limits, tests Details against representative parent rows, and tests cross-source enrichments using the same common-field normalization rules used at runtime. Direct database connectors group candidate equality values for the same child field as read-only `OR` alternatives and apply the row limit after filtering. Feed/file/custom API enrichments use a bounded local lookup scan where portable server-side equality filtering is unavailable; OData and supported ecommerce APIs may push compatible filters to the service. If a schema, field, filter, relation, or enrichment is repairable, Cifru sends one correction request and retests the result. The repair request contains the previous declarative JSON, safe diagnostics, and, only when enabled and necessary, at most one identifier-only mapping sample per relevant object. Authentication, permission, network, TLS, and timeout errors are shown directly and are not sent to AI for repair.

Provider testing is deliberately separate from connector testing. On July 11, 2026, the synthetic OpenAI evaluation ran 68 novice-style prompts across all 17 Cifru source kinds with `gpt-5-mini-`

`2025-08-07`. All 68 responses contained the required list/Details/enrichment/widget topology, but 30 raw responses also contained at least one unrequested relation, enrichment, or widget. This is why local scope normalization is mandatory. These synthetic provider tests do not claim that every third-party server, permission model, plugin, or schema is live-compatible; the generated draft must still pass the on-device production-connector dry-run before Apply becomes available.

The AI JSON can describe `lists`, `homeScreens`, `relations`, `enrichments`, and `widgets`. This allows a natural prompt such as: "For my ERP, create latest supplier invoices, add Details for invoice lines, and show product/feed data on invoice lines by matching EAN." Cifru maps this to native Lists, Home buttons, row Details buttons, and card enrichment fields after validation and user confirmation. For XML competitor feeds with fields such as `EAN`, `LinkDedeman`, and `LinkMathaus`, the expected setup is invoice lines as the displayed parent list, the XML feed as the lookup child list, a common product/EAN field as the match, and Link* fields as display fields.

Public manuals are deliberately not copied into built-in completion rules. The same compiler handles an ERP, ecommerce platform, ticketing system, public API, spreadsheet, feed, or an unknown internal database. Research may tell the external provider what a business concept usually means and which public object names to search for, but Cifru selects only objects and fields present in the configured source. This avoids a fix for one documented application becoming a hidden exception that makes the other flows less reliable.

> **Important safety rule:** Cifru does not execute AI-generated code. The AI Configurator cannot download modules, create plugins, run scripts, evaluate code, or add behavior that is not already implemented in the app binary. It can only request existing widgets, filters, actions, Home/dashboard objects, and read-only source metadata.

Area	Control	Type	Purpose
Request	Describe what you want to see in the dashboard	Text editor + example buttons	User writes a natural language request or starts with List with Details, Today's total, Top 10, or Last 7 days.

Area	Control	Type	Purpose
Request	Which sources to use	Picker	Defaults to Automatically choose relevant sources. A user may choose one configured list/connection. Cifru resolves connections locally to concrete readable objects. Initial generation sends only object/list IDs, display names, field names/types, and allowed capabilities; optional bounded identifier samples are reserved for automatic repair and can be disabled.
Request	Application/system to integrate	Text field	Name the app, ERP, ecommerce platform, feed vendor, or internal system you want Cifru to integrate. You may refer to a configured source by its exact name/alias, for example "use WooCommerce B2B"; Cifru prioritizes safe metadata belonging to that alias and keeps same-type sources distinct. Also explain which buttons/screens/widgets should be created. Do not put credentials here.
External AI Documentation Research	Collapsed research controls	Disclosure + approval toggle	Optional and shown only when external research is enabled in Settings with OpenAI, Claude, or Gemini. Manual URL/text fields remain nested until requested. URLs must be public, not local/private/internal.
External AI Documentation Research	User-approved public documentation text	Text editor	Optional documentation excerpt pasted by the user. It must not contain private credentials, real feed rows, invoices, customers, prices, stock values, tokens, or private endpoints.

Area	Control	Type	Purpose
External AI Documentation Research	Approval toggle	Toggle	User confirms the documentation is public and approves sending it to the external provider. Research remains optional: leaving this toggle off never blocks schema-only draft generation. After the user has enabled research globally and accepted its privacy notice, the per-request toggle starts enabled when the research control is available; the user can still turn it off before generating.
Actions	Generate draft / Stop generation	Stateful highlighted button	The enabled next-step action has a subtle shimmer, respecting Reduce Motion. It runs the full pipeline: discover schema, build safe context, optionally research approved manuals, request JSON, validate, dry-run, repair once when appropriate, and retest. While active it clearly says that generation may take a few minutes and becomes a real Stop action. iOS background execution is finite and best-effort; external requests use a 240-second timeout. Missing prerequisites and failures produce a visible status.
Actions	Current step	Single progress status	Shows one current operational step, such as discovering schema, waiting for declarative JSON, testing read-only queries, repairing mappings, or retesting. It is not model chain-of-thought and does not keep a long trace history.
Actions	Regenerate / Restore previous draft	Buttons	Generates another version without losing the immediately previous draft; the two versions can be swapped for comparison.

Area	Control	Type	Purpose
Actions	Discard draft	Button	Discards the current preview; this is separate from Stop generation.
Preview dashboard	Visual Home cards, nested Details, enrichments, and widgets	Human-first visual preview	Appears first and shows the layout/parent-child structure Cifru will create without exposing real business values.
Change summary	What will change	Summary rows	States how many lists, Home buttons, Details actions, cross-source completions, and widgets will be added or updated and states that unrelated workspace items are not deleted.
Technical preview	Technical structure + JSON	Collapsed disclosures	Shows the relation graph and exact declarative JSON only when the user expands them.
Automatic verification	Functional check list	Pass/warning/fail labels	Shows the result for each list, widget, Details relation, and cross-source enrichment. Apply remains disabled while any failed check is present.
Documentation used and assumptions	Manual titles, public URLs, uncertainty	Collapsed selectable list	When research is used, shows which public manuals informed the draft and which claims still need user verification without flooding the main screen.
Warnings and errors	Validation list	Collapsed notes + visible errors	AI notes are collapsed by default. Validator and functional errors remain visible and block Apply configuration.

Area	Control	Type	Purpose
Apply	Apply configuration + Undo last AI apply	Confirmation and restore buttons	Creates or updates native elements only after validation, functional verification and manual confirmation. Reapplying the same preview is disabled. Before mutation, Cifru snapshots the local workspace so the last AI apply can be undone. Undo restores local lists, screens, relations, enrichments, widgets and cache; it never writes to a configured source.

Allowed AI JSON Shape

The AI should return a single JSON object, not Markdown. Cifru expects this shape:

```
{
  "version": 1,
  "dashboardTitle": "Sales and stock",
  "dataSources": [{"id": "safe_internal_id"}],
  "widgets": [
    {
      "type": "metric",
      "title": "Total sales",
      "dataSourceId": "safe_internal_id",
      "field": "totalAmount",
      "aggregation": "sum",
      "filters": [{"field": "date", "operator": "last_days", "value": 30}]
    }
  ],
  "filters": [],
  "transformations": [],
  "actions": [],
  "warnings": [],
  "documentationAssumptions": [],
  "requiresUserConfirmation": true
}
```

- Allowed widget types: `metric`, `table`, `line_chart`, `bar_chart`, `pie_chart`, `alert_list`.
- Allowed actions: `refresh_data`, `open_detail_view`, `apply_filter`, `clear_filter`, `export_view`, `navigate_to_dashboard`, only when the app already has that behavior.

- Allowed aggregations: `count`, `sum`, `average`, `min`, `max`, and `latest` / `current` for one sorted current value, depending on the field type and source capability.
- SQL, if present in a future declarative draft, must be read-only `SELECT` / `WITH`. Cifru blocks `INSERT`, `UPDATE`, `DELETE`, `DROP`, `ALTER`, `EXEC`, `TRUNCATE`, `MERGE`, `CREATE`, `GRANT`, and `REVOKE`.
- Cifru blocks JSON keys suggesting executable behavior: `code`, `script`, `eval`, `plugin`, `executable`, `runtime`, `module`, and `downloadedCode`.

Provider Choices

Both AI entry screens show provider readiness before generation. Apple On-Device and Mock / Local Templates do not require an API key and cannot browse the web. OpenAI, Claude, and Gemini are external providers and require a user-supplied key stored in Keychain; when the selected external provider has no saved key, Cifru shows a direct link to AI settings.

Provider	Use	Privacy / limits
Apple On-Device, Recommended	Default when Apple Foundation Models are available on supported iOS versions.	No API key. Runs on device. No web browsing. Uses prompt, safe metadata, schema, allowed widgets/actions, and local/user-provided documentation text.
Mock / Local Templates	Fallback for simulator, unsupported devices, unavailable Foundation Models, or testing without API.	No external provider. Uses built-in templates for sales, stock, top products, invoices, orders, and cashflow-style requests.
OpenAI	Optional external provider configured by the user.	API key is stored in Keychain. External documentation research is allowed only when enabled and approved by the user.
Claude	Optional external provider configured by the user.	API key is stored in Keychain. External documentation research is allowed only when enabled and approved by the user.
Gemini	Optional external provider configured by the user.	API key is stored in Keychain. External documentation research is allowed only when enabled and approved by the user.

For OpenAI, Cifru parses the raw Responses API output defensively. It does not assume that text is always located at `output[0].content[0].text`, because the response can include reasoning,

web-search, or other items before the final message. Cifru reads supported text/JSON output shapes and reports clearer errors for incomplete responses or output-limit issues.

Safe Metadata Sent To AI

Cifru constructs `SafeDataSourceMetadata` from configured lists. It may include:

- internal list/source id;
- display name;
- source type such as `sql_readonly`, `rest_api`, `ftp_feed`, `local_feed`, `vendor_api`, or `readonly_feed`;
- field names, field types, nullable flag, semantic/common field role;
- allowed operators, aggregations, widgets, actions, and capabilities.

Initial generation does not include source rows. If minimal mapping samples are enabled and automatic repair needs value-shape evidence, Cifru may include at most one reduced row per relevant object, limited to 12 bounded identifier-like fields such as document IDs, product codes, EAN/barcodes, or SKUs. Local filtering excludes credentials, secrets, tokens, URLs, hosts, usernames, customer/supplier/contact/name/address fields, descriptions/notes, prices, totals, stock, quantities, exports, and private/internal documentation URLs. Sample values are never copied into the saved declarative configuration, warnings, logs, or analytics.

External AI Help Chat And Copyable Prompt

The empty Home screen, Start guide, Help, AI Configurator, and the "Ask an external AI with Cifru" Siri/App Shortcut identify this surface separately from Cifru's built-in AI Configurator. If no OpenAI, Claude, or Gemini key is stored, it shows the localized copyable prompt below. If at least one external key is stored, Help opens a conversational assistant using the same provider model, base URL, and Keychain credential already configured in Cifru.

The external Help chat keeps a bounded transcript locally with iOS complete file protection and offers a New conversation action that deletes it. Every request starts from a vendor-neutral map of the controls and navigation paths implemented in the installed Cifru binary, plus generic recipes for authentication, latest-value screens, Details, cross-source matching, cache, and external widgets. It also retrieves relevant sections from a locally cached/indexed copy of the complete public Cifru HTML guide. Public web-manual search is disabled by default and is used only after the user explicitly enables it; public manuals supplement the installed UI map and cannot invent a control that Cifru does not contain. Only a bounded recent message window, safe source metadata, and relevant public documentation excerpts are sent after first-use confirmation. Common API keys, bearer tokens, password/connection-string fragments, and private/internal

URLs are redacted locally before transmission. The chat provides explanations only: it cannot create, apply, or silently modify a dashboard. It displays operational status such as reading documentation or waiting for the provider and never exposes hidden model chain-of-thought.

The copyable prompt is localized by app language and points to the HTML documentation. The English version is:

```
Hi! I want to integrate my app with Cifru and I need your help.

The Cifru documentation is here: https://cifru.eu/documentation.html
The public Cifru configuration library is here: https://cifru.eu/community

Please review the Cifru documentation and the public online documentation for the
app I want to integrate. Tell me:
1. which integration types I can use in Cifru;
2. which data I can bring in read-only mode;
3. which APIs, feeds, tables, endpoints, or files are useful;
4. how to configure everything in Cifru step by step, in detail, for a non-
programmer;
5. which fields should be mapped;
6. which same-card fields from other lists or row Details links would make sense.

The app I want to integrate is: X
```

Privacy, Legal, And Responsibility

The user chooses and controls their own sources. The user is responsible for credentials, permissions, third-party service terms, source availability, source-side access controls, and decisions made from displayed data. Cifru is not a backup system, accounting authority, audit system, legal system, medical system, financial trading system, or source of truth.

Use read-only accounts, least-privilege permissions, TLS/HTTPS, VPN or private network access where appropriate, and independent source-side logging for important systems.

Related pages: Cifru Home - Documentation page - Configuration library - Privacy Policy - Terms of Use - Support