

RELEASE NOTES

Dawiso 2026.3 LTS

Long-Term Support Release

September 2026

Contents

Administration & Integrations.....	4
Checkout Routing.....	4
Data Source Watching.....	5
Metamodel Admin.....	6
Package Installation.....	6
AI & LLM Support.....	7
AI Chat.....	7
Model Context Protocol.....	8
Tool Management.....	8
API Tables Improvements.....	10
Filters.....	10
Manage Columns.....	11
Table Row Count.....	11
Automation.....	11
Expressions.....	12
Triggers.....	12
Customizable Layouts Using Widgets.....	12
Homepage & My Customizations.....	12
Dashboard Authoring.....	14
Object Detail Layout.....	15
Dashboards.....	16
Page & Dashboard Permissions.....	16
Data Product Standard.....	17
Data Market.....	18
Data Product Creation.....	18
OSI Export.....	19
Diagrams.....	20
AI Diagram Creation & Editing.....	21
Drawing & Annotation.....	21
Entity Relation Diagrams.....	22
Knowledge Graph.....	22
Lineage.....	23
External Link Checking.....	24
Import/Export Improvements.....	24
Export Reliability.....	25
Sample Data.....	25
Interface Redesign.....	25
Login Page.....	26
Users & Permissions.....	27
Objects.....	28
Change Object Type.....	28

Hierarchy & Inheritance Reliability.....	29
Object Numbering.....	30
Workflow Transitions.....	31
Onboarding & Guided Tours.....	31
First-Login Walkthrough.....	31
Guided Tours.....	32
Onboarding Checklist.....	32
Onboarding Hints.....	32
Packages.....	33
Attribute Validation.....	33
Metadata Sync.....	33
New Components.....	33
Package Editor.....	34
Packages Updates.....	35
Packages: Marketplace Improvements.....	36
Enabled by Default.....	36
Reserved core Prefix.....	36
Prerequisite Resolution.....	37
Packages: New data Type.....	37
Automatic Installation.....	37
Export Object Data.....	37
New Schema.....	37
Product Tours and Catalog Objects.....	38
Security & Permissions.....	38
Unified Platform.....	39
Gate.....	39
Identity Service.....	40
Platform Performance Improvements.....	40
UX/UI Enhancements.....	41
Editor.....	42
Fullness Score Detailed Overview.....	42
Navigation.....	43
NPS Survey.....	43
Spaces Icons.....	43
Tooltips Improvement.....	44
User Relation Actions.....	44

This Long-Term Support (LTS) release note for Dawiso 2026.3 consolidates all platform updates delivered since Dawiso 2026.2. The whole platform was redesigned and optimized for a cleaner, faster experience, alongside major new capabilities: Data Product Standard, a new application for describing and sharing data as governed, ready-to-consume products, and Unified Platform, connecting Dawiso to Gate, a new external app, through a shared identity service.

Administration & Integrations

This release adds a set of new administrative settings, covering checkout routing, data source monitoring, metamodel visibility, and package installation safety.

Checkout Routing

Requesting access to a data product goes through a cart-style checkout process. Cart checkout can now route to Jira Service Management or ServiceNow instead of Dawiso Gate. In Jira Service Management, each product becomes its own customer request; in ServiceNow, each one becomes a record in a table you choose. Switching between Gate, Jira Service Management, and ServiceNow takes effect immediately, with no restart needed.

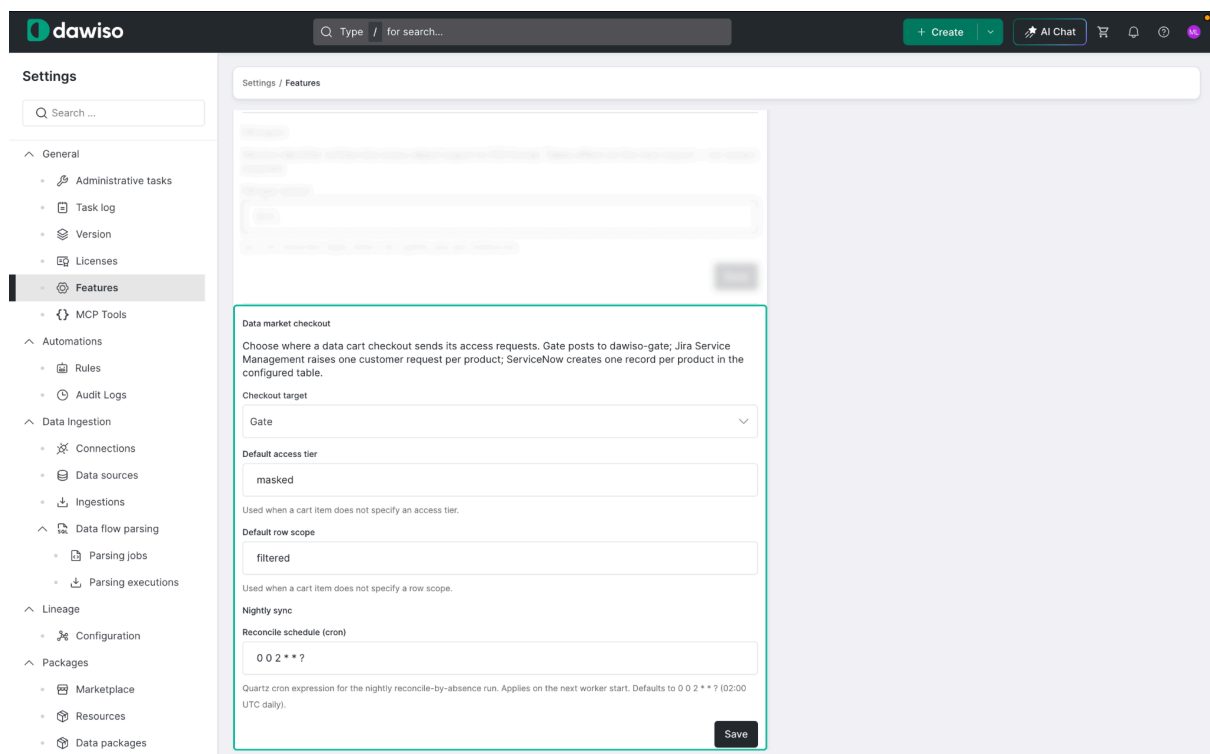


Image 1: Checkout target picker set to Gate

Switching targets doesn't carry over requests already raised. Request status only flows back into Dawiso for Gate; for the other two targets, **My Access** shows an explanatory note in place of a status list. See [Data market checkout targets](#) for the full configuration reference.

Data Source Watching

Administrators can now watch a data source to be told when its ingestions fail, instead of checking the ingestion list for problems.

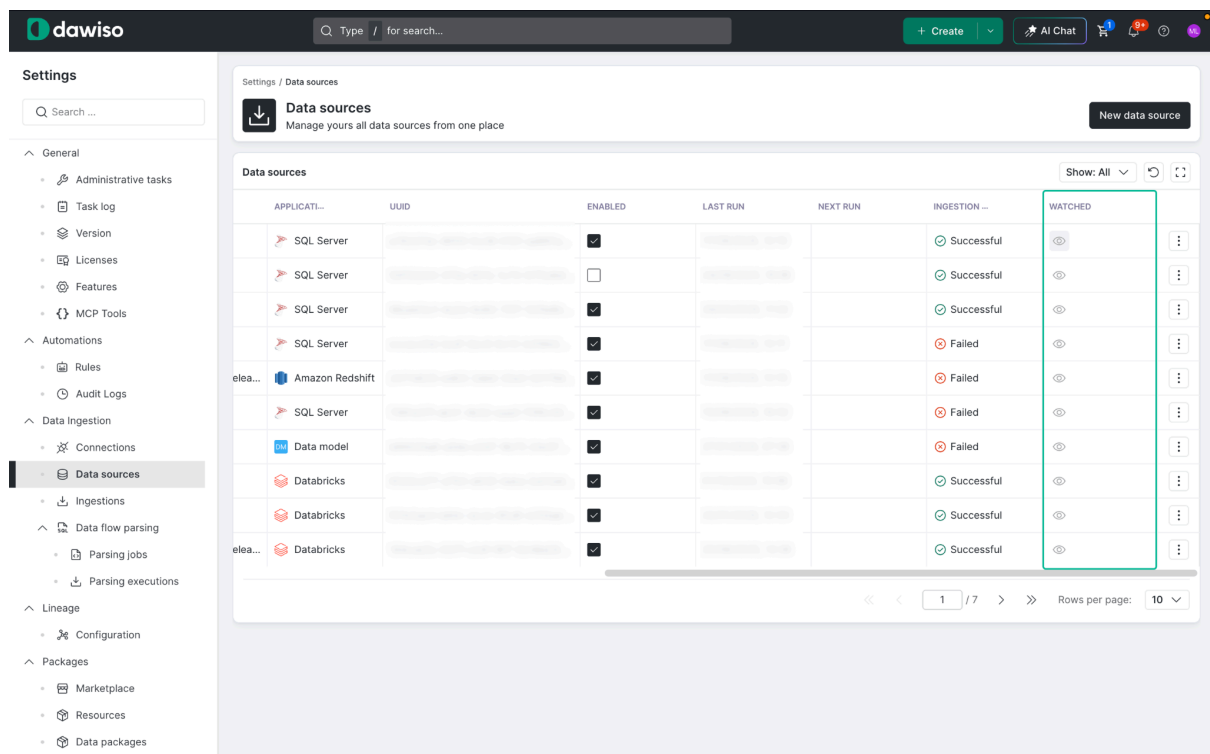


Image 2: Data sources list with a Watched column of eye icons

A newly created data source is watched by its creator right away, while one that predates this feature stays unwatched until an admin opts in. A failed ingestion run notifies every watcher by email and in the notification bell's **Watching** tab, naming the data source, the failure state, and when the run started and finished.

Canceled runs don't notify, and a run stuck reporting the same failure repeatedly only notifies once. See [Watching data sources](#) for the full reference.

Metamodel Admin

Metamodel Admin adds a read-only, searchable inventory of an instance's whole metamodel: applications, types, attributes, relations, domains, and workflows, all in one place. Finding which object types reference a given attribute, for example, no longer means querying the database directly; it's a filter away in the new browser, available to administrators. See [Metamodel Admin](#) for the full reference.

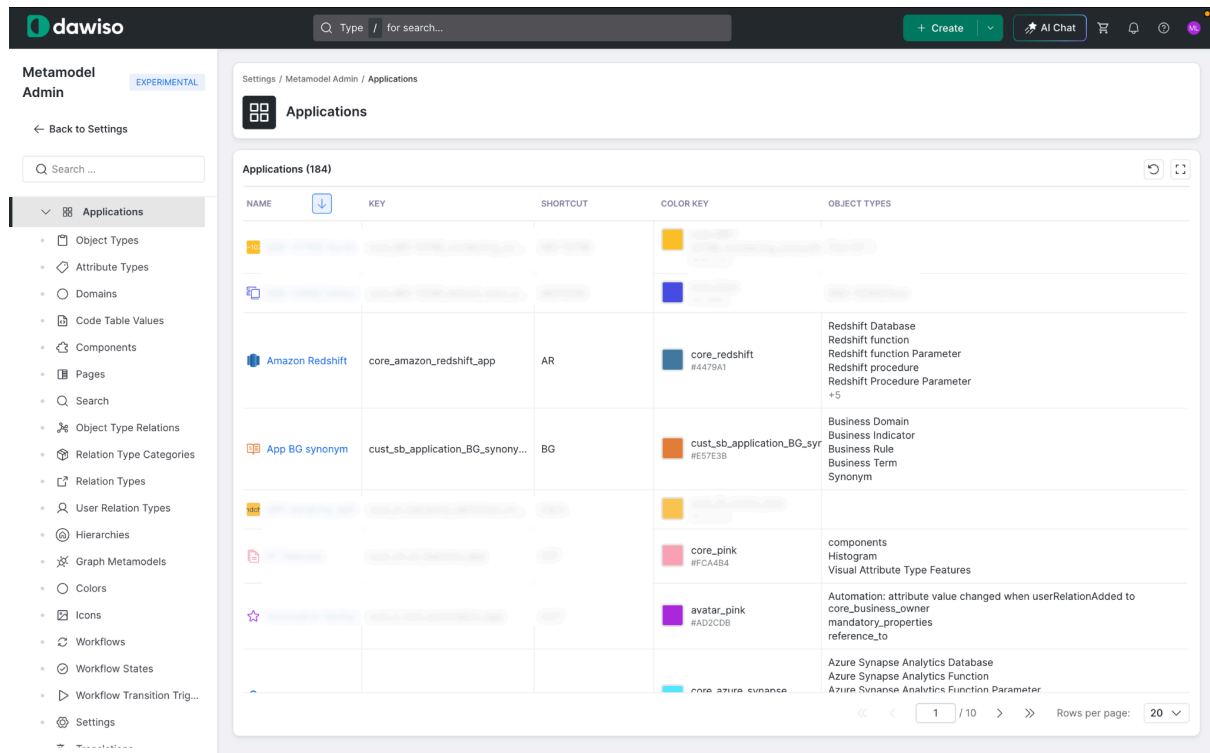


Image 3: Metamodel Admin's Applications screen

Package Installation

Package installs got both safer and more automatic:

- **Destructive-patch validation:** Installing a patch that would silently drop existing metamodel configuration now fails validation instead of completing unnoticed. See [Patch packages — Advanced Guide](#) for the technical detail.
- **Automatic metamodel propagation:** Installing a package now propagates its metamodel changes automatically, removing the need for a separate manual template refresh.

- **REST endpoint for user sync:** User sync can now also read organizational structure from a REST endpoint, alongside its existing input options.

AI & LLM Support

Two changes reshaped AI and LLM support. A new **AI Chat** panel ships in early access, and the Model Context Protocol tool surface was rebuilt into narrower, role-filtered tools.

AI Chat

AI Chat is now easily available from the top navigation bar, where you talk to Dawiso AI about your catalog. This feature is in early access, and token usage is currently free. Continuing to use it once early access ends will require a license update.

Responses stream in as Markdown, with clickable citations linking straight back to the objects they're drawn from. Minimizing the panel or refreshing the page doesn't end the conversation, and each answer can be rated with a thumbs up or down.

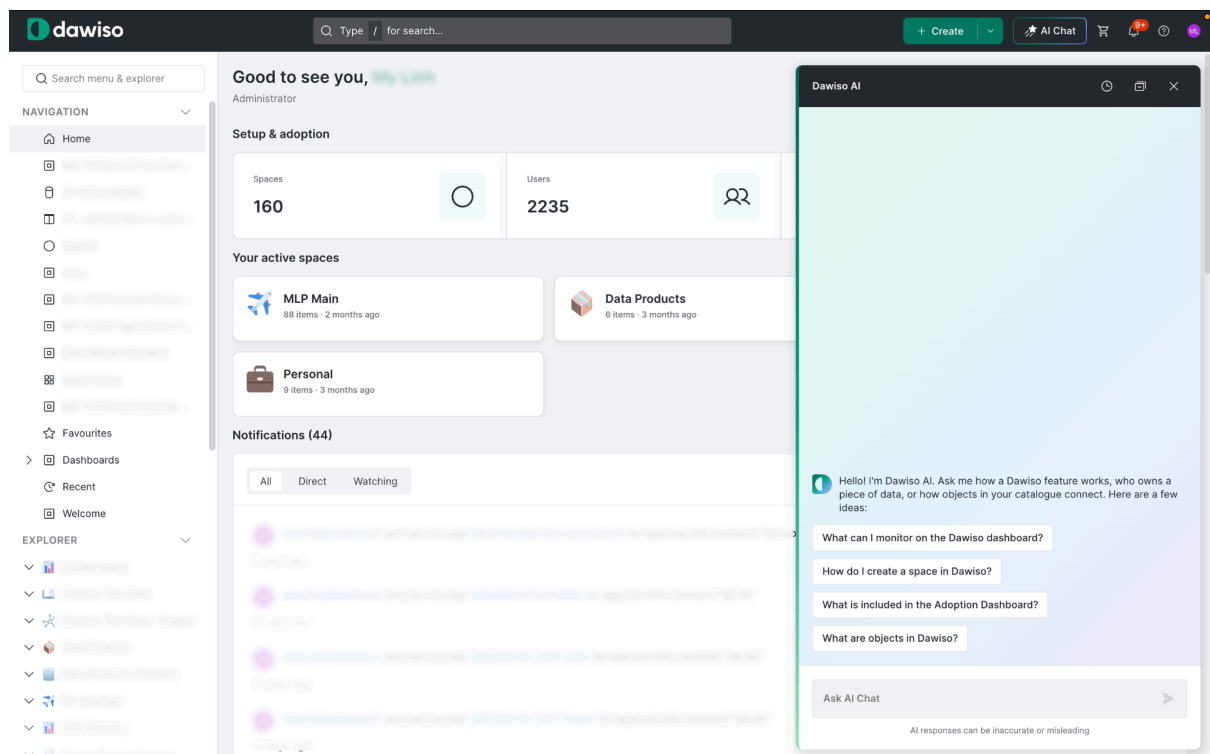


Image 4: AI Chat panel open over the Dawiso dashboard, with suggested starter questions

Every query is limited to spaces the asker already has read access to. An object outside that scope never surfaces in a response or its citations.

Model Context Protocol

MCP's broad, multi-action tools were rebuilt into narrower, role-filtered ones, each handling exactly one kind of read or write. Calling `tools/list` from a **Viewer**, **Contributor**, or **Admin** session no longer returns the same catalog; each role only sees the tools it's allowed to use. AI agents already integrated using the previous tool names won't pick up the change automatically; each one has to reconnect so `tools/list` reflects the new set. Existing integrations should treat this as a breaking change. See [Dawiso Model Context Protocol \(MCP\)](#) for the full tool reference.

Additional improvements round out the MCP update:

- **Local user management:** An admin-only tool can now add, update, remove, or reset the password for local Dawiso users, for environments using in-memory authentication rather than SSO.
- **Single-call attribute updates:** `update_object` writes several attributes in one call instead of one call per attribute.
- **Object creation without a concept:** `object_management` can create objects even for types that normally require an associated concept first.
- **Leaner responses:** Tool descriptions are shorter, with parameters documented in their schema, and every tool shares one response envelope. Results page at 20 items by default, capped at 100, with new `mode` and `fields` parameters to control how much comes back. See [Dawiso Model Context Protocol \(MCP\)](#) for the full reference.

Tool Management

Administrators now manage the environment-wide tool allowlist from a new **Settings > General > MCP Tools** page, with separate **Production** and **Lab** tool sets and two switches restricting Contributors to read-only tools or Admins to Contributor tools. From **My AI tools**, each person can narrow their own available tools within whatever the administrator allows, with sortable, filterable tool tables for both production and lab tools. See [MCP Tools Management](#) for both settings pages in full.

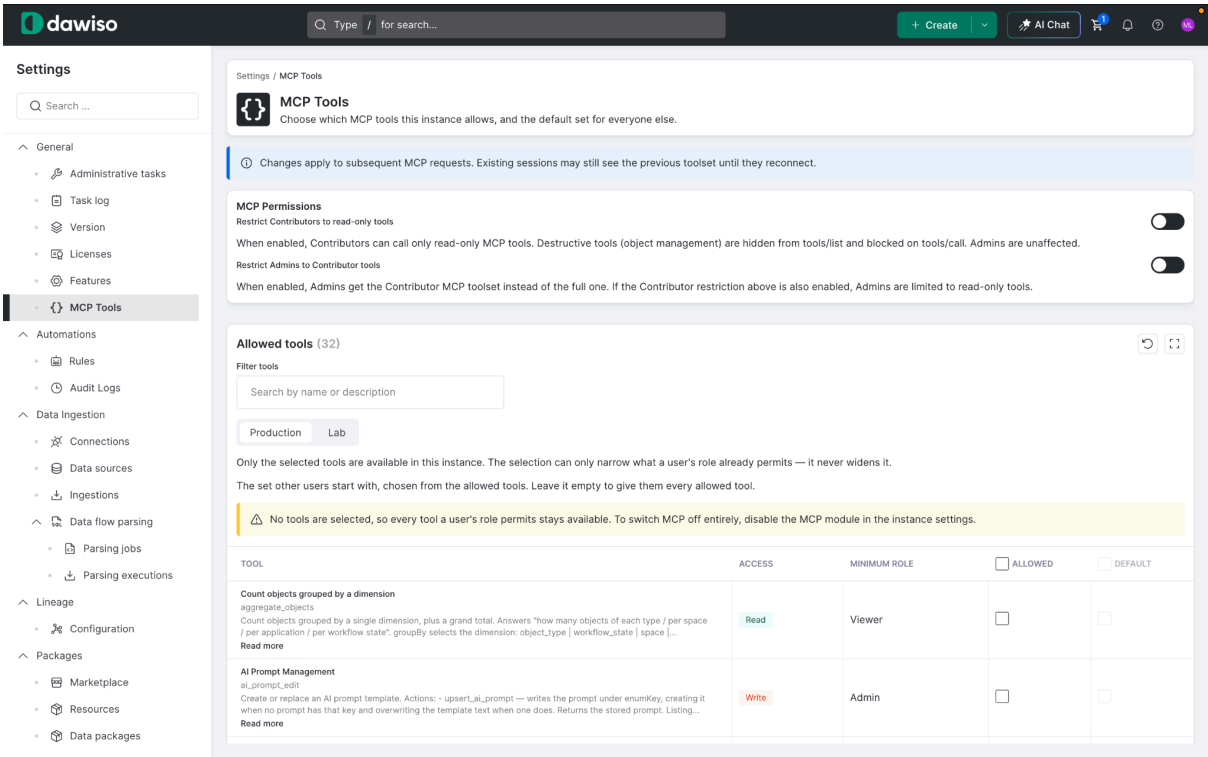


Image 5: MCP Tools settings page with the MCP Permissions switches and the Allowed tools table

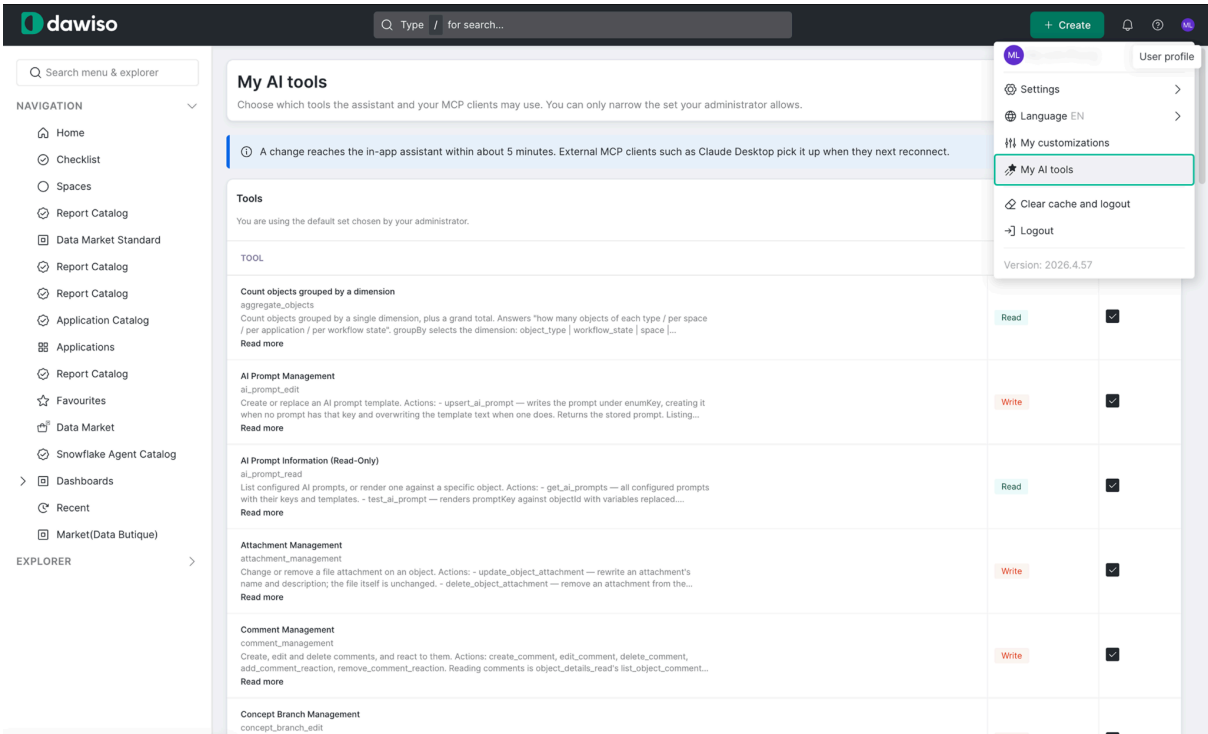


Image 6: My AI tools page listing available tools with their Read or Write access and an Add checkbox

API Tables Improvements

Tables built with the API cards and API tables component gained more control: choose which columns show, filter numeric columns by range, and set a different starting row count.

Filters

Table filters gained three separate improvements:

- **Range filters on numbers:** Aggregated numeric columns, such as karma or view counts, can be filtered by a **from-to** range instead of an exact value. See [API cards and API tables](#).
- **Saved filters:** A filter can be named, saved, and reused, and whichever one you used most recently loads again automatically when you reopen the table.
- **Searchable relation filters:** A relation column's filter is now a searchable multiselect instead of free text: open it to load the related objects, search by name, and scroll for more as they load in. Picking several objects in one column matches rows with any of them, and picking objects across two different relation columns narrows to rows matching both. **Select all** only grabs what's already loaded, so scrolling to the bottom first is still necessary to select everything.

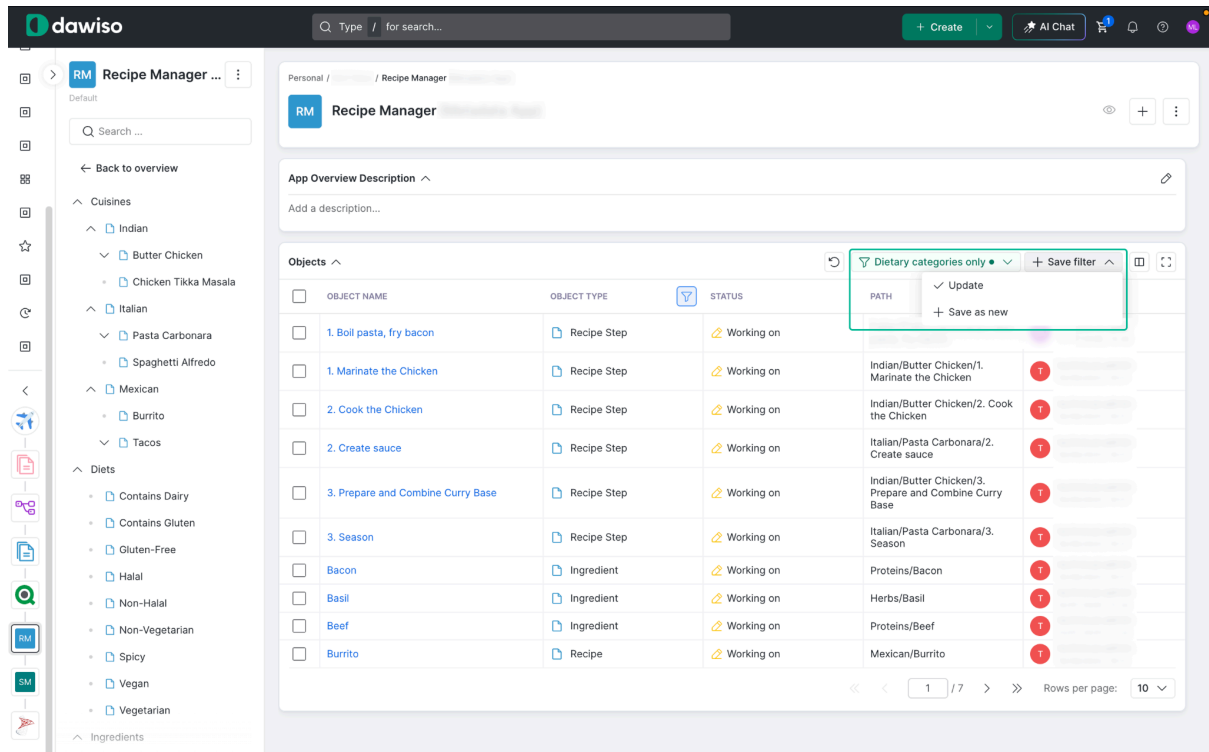


Image 7: A saved filter's dropdown, showing Update and Save as new options

Manage Columns

Any table listing Dawiso objects now has a **Manage columns** control built into it, for hiding, reordering, searching, and showing columns without leaving the view. Metamodel attributes and user relations can be added as columns too, beyond the table's default set. **Restore default columns** reverts any customization back to how the table started. See [API cards and API tables](#) for the full reference.

Table Row Count

Any table built with the API cards and API tables component, dashboards included, can start out at a page size the table's author chose instead of a fixed default. Once a user picks their own rows-per-page value, that choice sticks and overrides the configured one from then on. See [API cards and API tables](#) for the full reference.

Automation

Automation triggers now catch more indirect changes, and dynamic expressions gained a formatting helper for rich text.

Expressions

The `.stripHtml()` function is available in dynamic expressions. It strips HTML tags from a value and collapses it into single-line plain text. Use it when passing a rich text attribute into `append()` or `appendLine()`; without it, the attribute's `<p>` tags break the line. See [Functions](#) for the full list of expression functions.

Triggers

The `userRelationAdded` and `userRelationDeleted` triggers now also fire on role changes at the space level, on inheritance to descendants, and on workflow transitions. Previously, only a direct edit fired them. Existing automation configuration is unaffected.

Customizable Layouts Using Widgets

Dawiso's homepage, dashboards, and now individual object detail pages all let you rearrange their content directly, instead of relying on a fixed layout. The homepage and dashboards use a free-canvas grid of widgets that can be added, moved, and resized; object detail pages use a similar drag-and-drop approach within a fixed two-column structure. A dashboard's widget catalog, for example, includes a **Broken links** widget tracking external links that no longer resolve, covered under [External Link Checking](#).

Homepage & My Customizations

The left navigation menu works the same way in Customize mode, where items can be reordered or nested under one another by dragging a handle. Changes on either surface stay as an unsaved draft until you **Save**, with **Discard** and **Reset** available to back out.

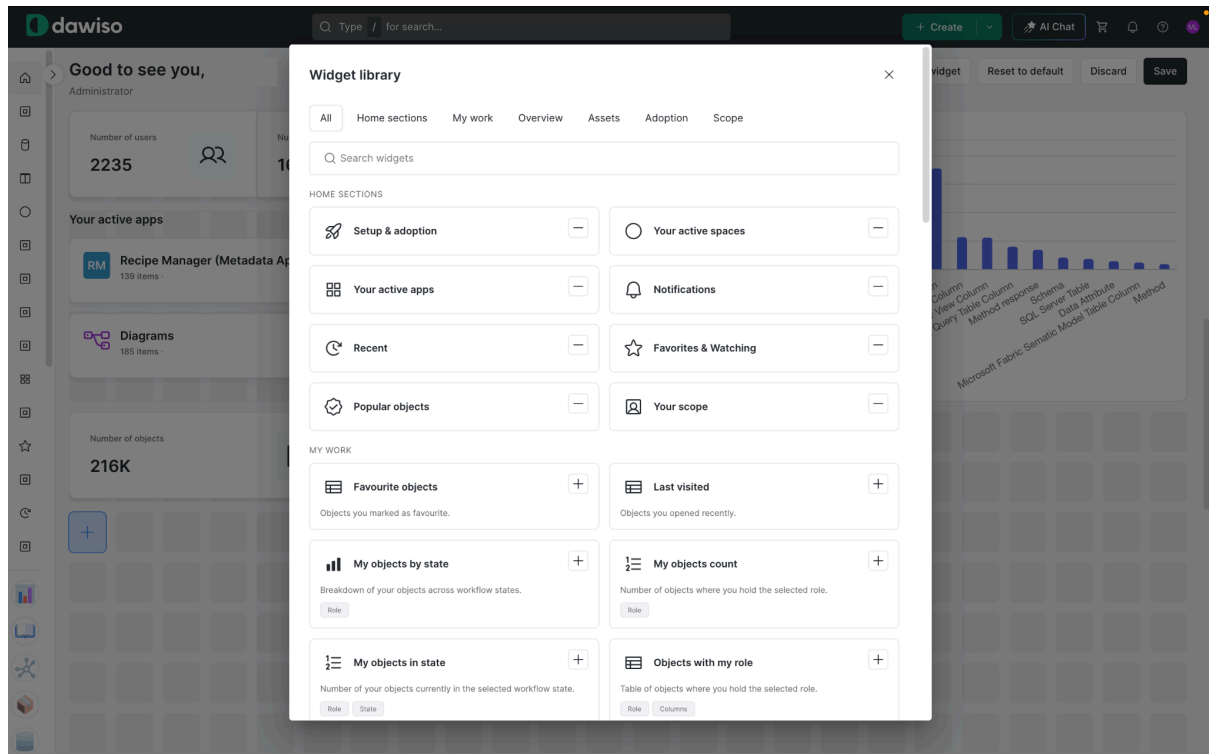


Image 8: The widget catalog opened from a homepage in Customize mode

See [Homepage layout and widgets](#) and [Customize navigation](#) for the full reference.

Administrators can publish a layout to everyone, or to selected license roles and groups:

- A homepage, navigation, or Space layout can be published this way.
- Each publication is versioned and can be previewed before anyone switches to it.
- Every user's **My customizations** screen lists what's available to them on each surface: their own arrangement, any published layout they're entitled to, or **Automatic**.
- A **Used by you** badge marks whichever one currently applies.
- A layout a user picks for themselves stays in effect even after a higher-priority layout is published, until they switch back to **Automatic**.

Copying and pasting a widget on the homepage now lands it wherever the pointer sits: on an empty cell, on top of another widget (pushing it and everything below down), or below the copied widget when pasted without pointing anywhere in particular. See [My customizations](#) and [Published customizations](#) for details.

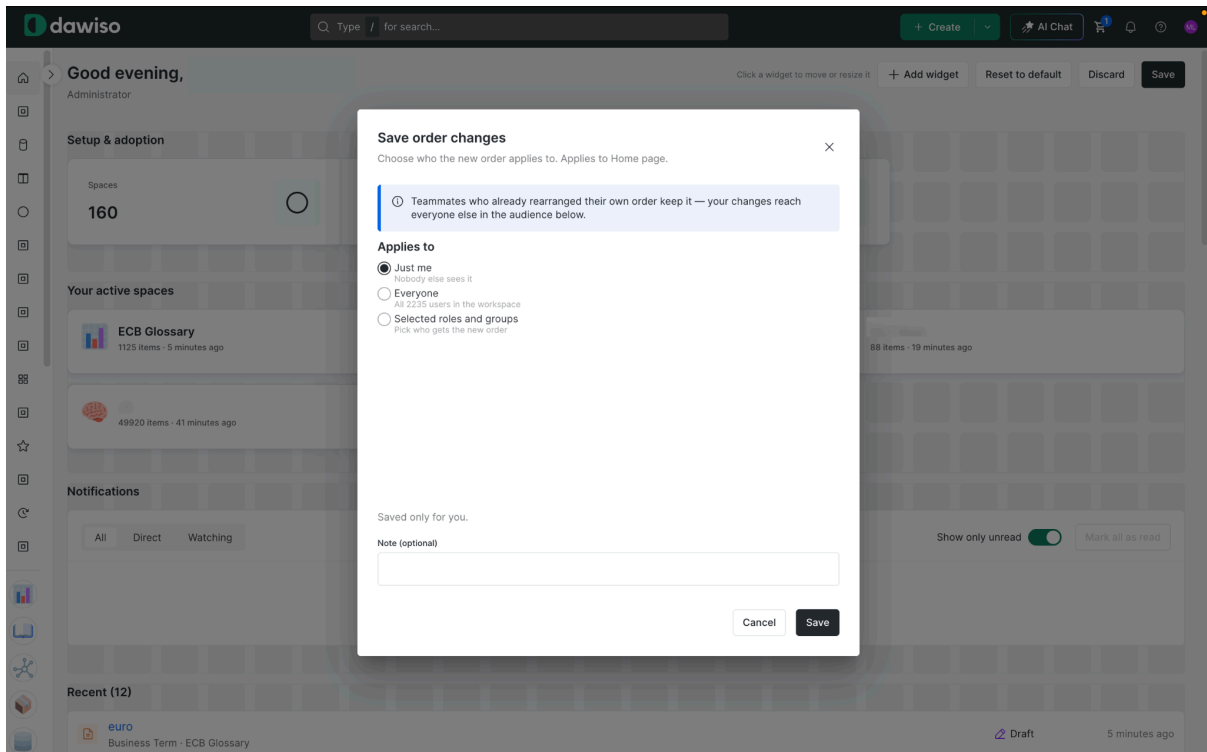


Image 9: Save order changes dialog with Applies to options: Just me, Everyone, and Selected roles and groups

Dashboard Authoring

Administrators can create a named dashboard from **Customize** mode. Saving it creates an installable package behind the scenes, versioned on every edit, so the dashboard can move between environments like any other package. The dashboard becomes a page in the left navigation. Creating, editing, or deleting it still requires administrator permissions. Viewing it follows the dashboard's own page permissions. See [Dashboards](#) for the full walkthrough.

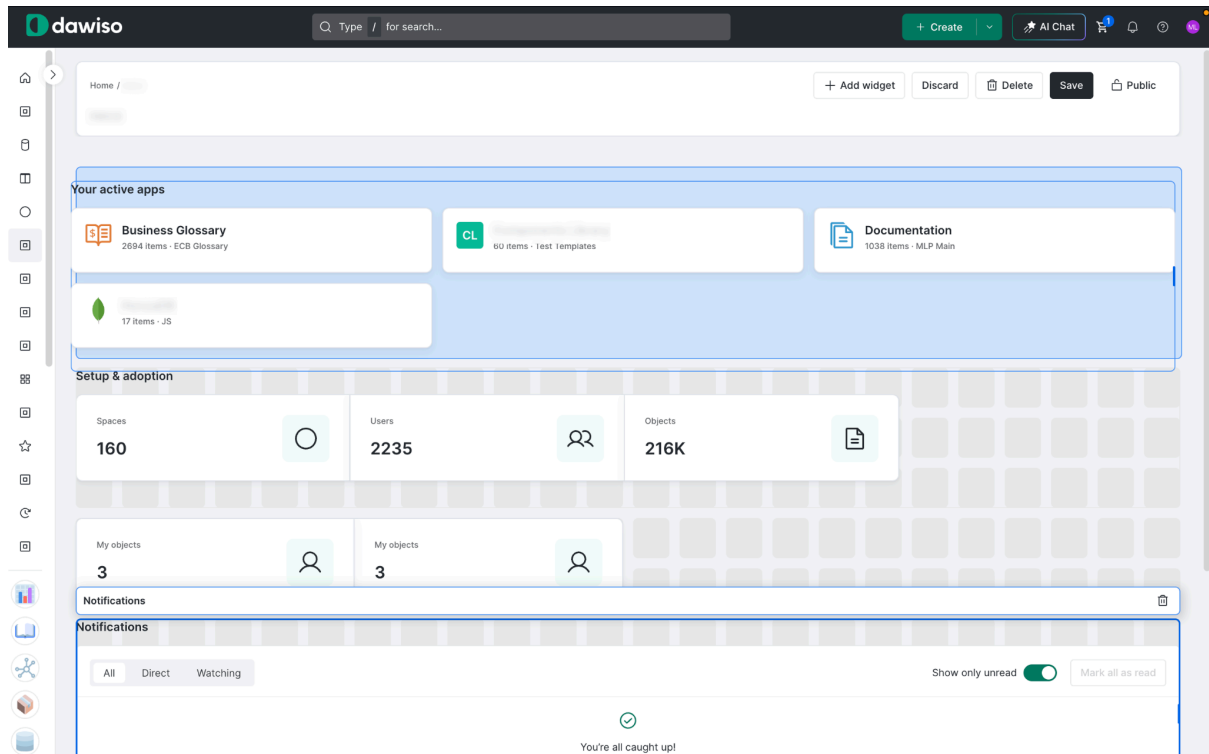


Image 10: A dashboard open in Customize mode, with Add widget, Discard, Delete, and Save controls

Object Detail Layout

Instead of hand-editing package files, administrators can now drag, add, and remove an object type's detail-page components right on a live object, with every change reflected immediately. Access it from the object's **three dots** menu via **Edit template components**; the result saves as the new layout for every object of that type.

Components can be added, moved, duplicated, renamed, configured, or removed, and dragged to a new position by their handle. Rows inside an attribute, relation, or related-users list can also move independently, either within their own list or promoted into a standalone component.

A few rules keep the layout coherent:

- Panels and diagrams only sit at the top level of a column; a relations table is the one exception that can also go inside a panel.

- Adding a component at a column's top level wraps it in a panel that needs its own name, except for a standalone rich-text attribute or diagram, since those already draw their own frame.
- Editing a shared component from a package for the first time detaches it into that object type's own copy, so later package updates no longer reach it.

See [Object detail layout](#) for the full reference.

Dashboards

Dashboard pages gained finer-grained permissions. For rearranging dashboard content itself, see [Customizable Layouts Using Widgets](#).

Page & Dashboard Permissions

Each dashboard page can be set to **Public** or **Private**. A **Private** page stays invisible to everyone, administrators included, until access is explicitly granted. Regardless of that setting, access can be granted per user or group at **Read** or **Read & Write** level. See [Dashboards](#) for the full permissions reference.

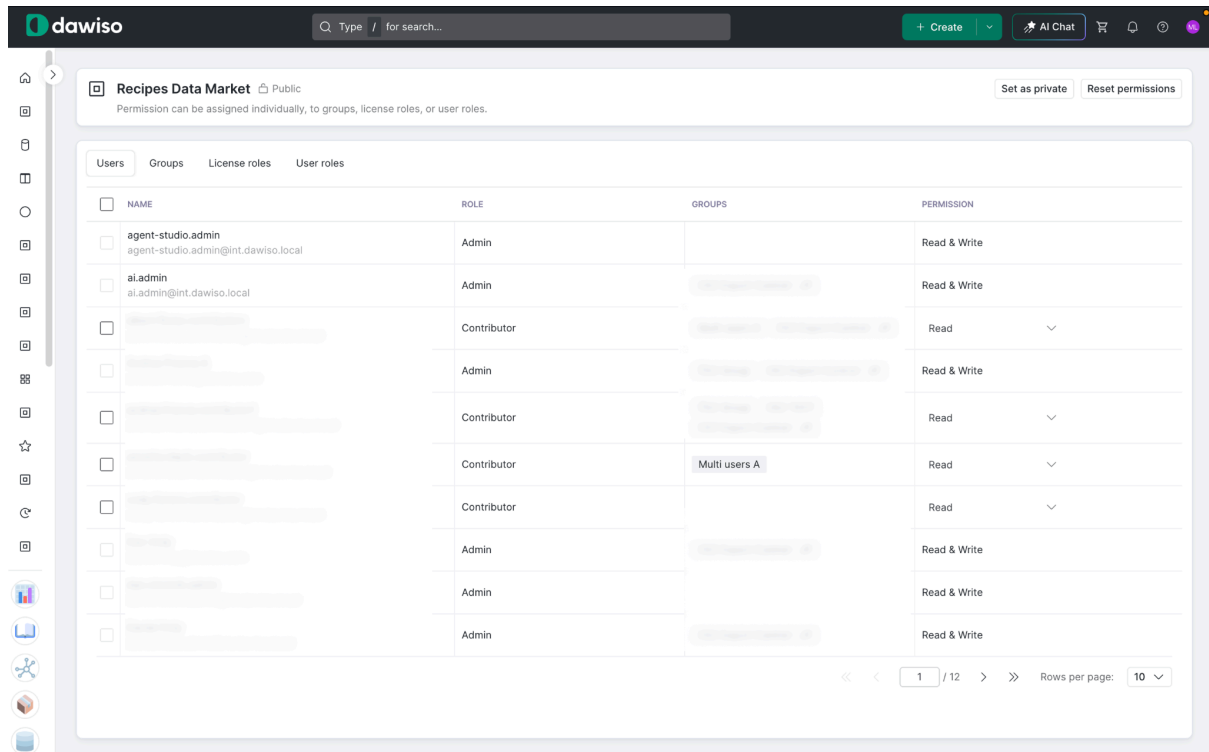


Image 11: A dashboard's permission screen listing users with their role and Read or Read & Write permission

Data Product Standard

Data Product Standard is a new application for describing, governing, and sharing data as trustworthy, ready-to-consume packages instead of free-form catalog entries.

Every product is assembled from the same typed pieces: ports for exposing or consuming data, a contract covering what's delivered and its schema, quality and service-level guarantees, and an accountable owner. None of this is a Dawiso invention. The whole model maps to the Open Data Product Specification, the Open Data Contract Standard, and Data Mesh port conventions, so a product described here exports cleanly to those formats too.

Products also connect to each other through their ports, and Dawiso draws each producer-to-consumer link as lineage, surfacing cross-team dependencies that would otherwise stay hidden.

Data Market

Once published, a data product surfaces in the Data Market for anyone in the organization to explore and formally request, instead of tracking down an owner informally. Each product also carries a visible classification for personal-data content and who's allowed to see it, so sensitivity is clear before anyone asks for access.

After requesting access, track its status from **My Access**, opened from the profile menu in the top right.

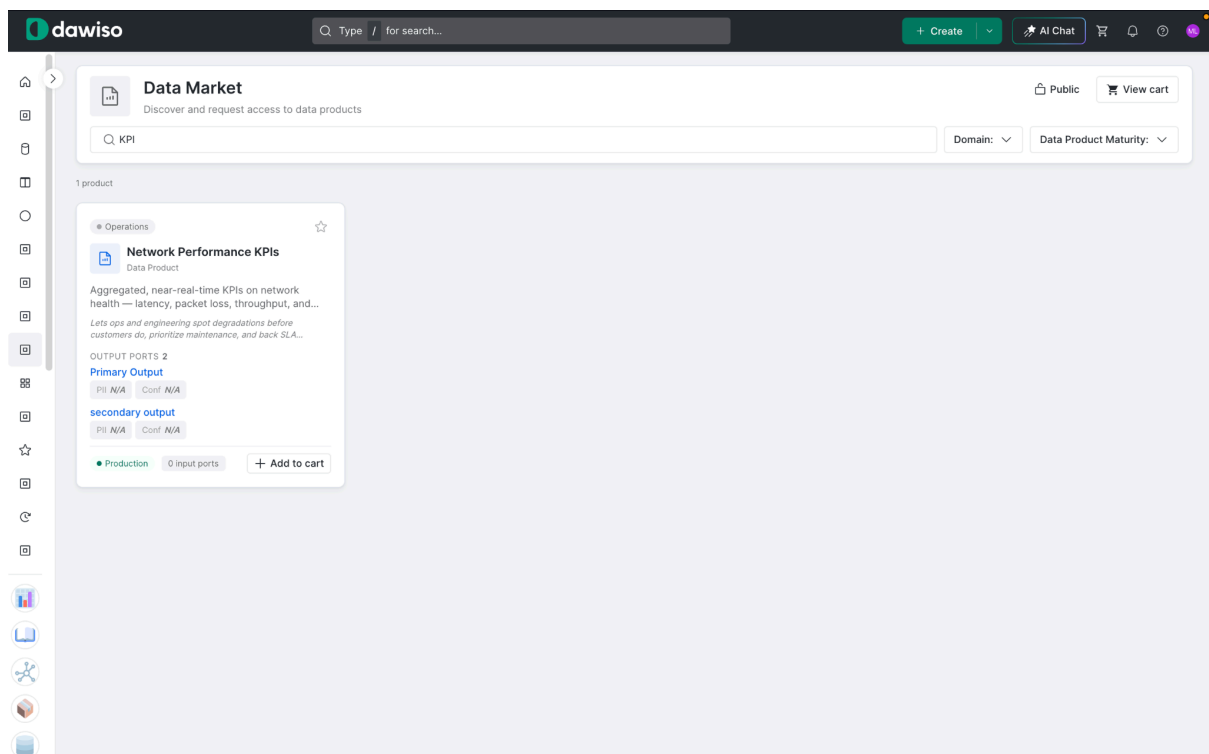


Image 12: Data Market search results for a data product, showing its output ports

Data Product Creation

Building a product is guided rather than freeform: a series of steps walks through identity, business context, ownership, the underlying data, and any contract terms, and won't let a product publish until it's actually complete. A **Business Context** step lets you attach the glossary terms, use cases, related products, and labels that make a product findable by meaning, not just by name.

Create data product

1 Basics 2 **Business Context** 3 Ownership 4 Data Assets 5 Ports 6 Contracts 7 Review

Business Terms & Indicators 1 attached

Glossary terms that define what this product means. Consumers search by these.

Search business terms and indicators...

Energy Starter Glossary / Business Glossary / Assets

Asset
Business Term - Draft

Use Cases + Add use case

Written here, not picked — each one becomes a child object of this product.

Use case name

Use case example

Description
Explain the goal, audience, and expected outcome.

Inter 12px Paragraph

Description of a use case.

Image 13: The Business Context step, with Business Terms & Indicators and Use Cases sections

An AI agent can also build a starting draft of an entire product from an object already in your catalog, prefilling identity, ownership, assets, ports, and a contract automatically. Reviewing, adjusting, and publishing that draft is still up to you, but starting from something prefilled removes most of the manual setup. Opening a prefilled draft this way replaces any product you already had in progress, without the usual prompt to resume or discard it, so finish or publish other in-progress work first.

Returning to an unfinished draft now blocks progress with a dialog to **Resume draft** or **Start over**, instead of a dismissible banner.

OSI Export

A data product can now be exported as **OSI YAML**, following the [Open Semantic Interchange](#) standard for tables, columns, and foreign-key relationships. The option sits next to **Export** wherever a data product is reachable: its own detail page, a multi-select in an app's or space's Objects table, or the **three dots** menu on an application or space.

An output port with no attached tables is left out of the file, and an object that isn't a data product, or whose ports were all skipped, is skipped entirely. What got left out is recorded in the export task's log rather than the notification.

The downloaded file works with any OSI-compatible tool, Snowflake included. For Snowflake, Dawiso can also generate a ready-to-run SQL file that loads the export directly. When Dawiso can't safely resolve a single target database and schema, or a data source isn't a plain table reference, it comments the statement out and explains why instead of guessing at a target.

Administrators can also change the specification version written into every export, which defaults to **0.1.1**. See [Export a data product as OSI YAML](#) for the full reference.

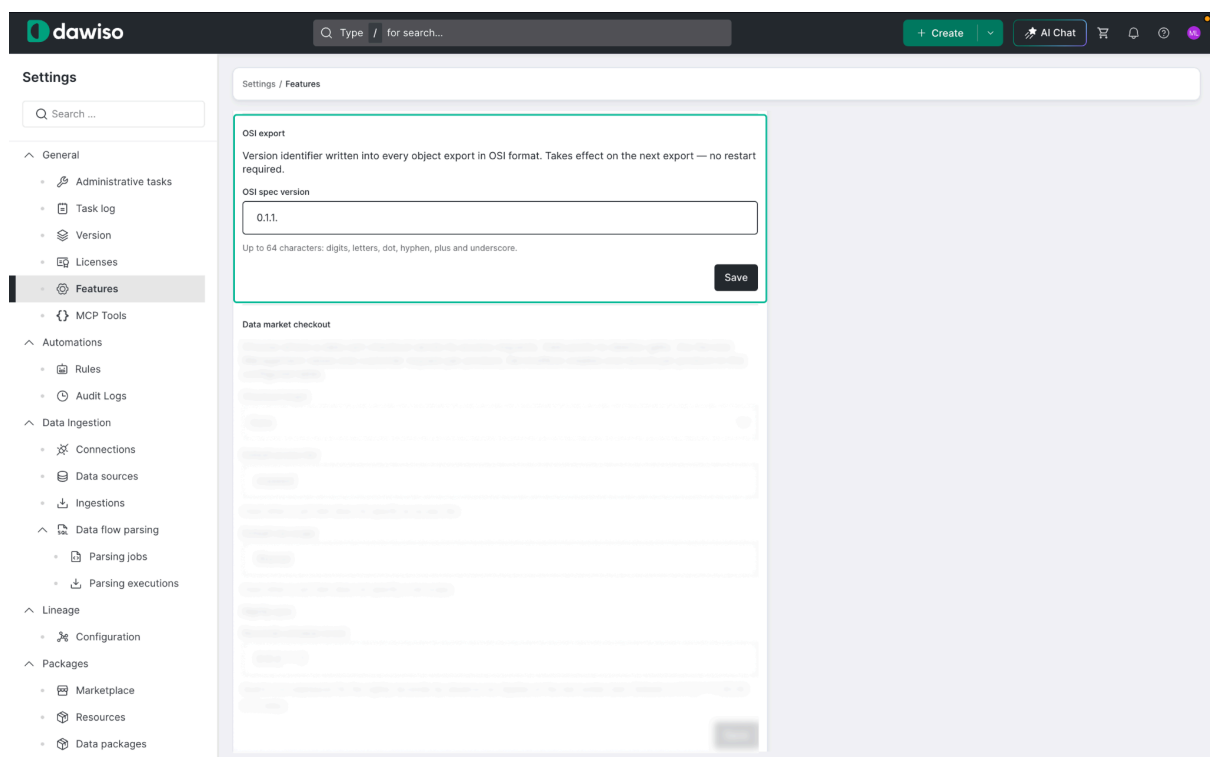


Image 14: OSI export settings showing the OSI spec version field

Diagrams

Diagrams gained a more capable AI assistant, freehand drawing and annotation tools, and a lineage layout built on ELK (Eclipse Layout Kernel, an open-source graph-layout engine) for large, complex flows.

AI Diagram Creation & Editing

AI-generated diagrams, introduced in 2026.2, now also edit an existing diagram instead of only creating a new one from a description. **Ask AI to edit** applies a plain-language instruction directly to the canvas — create, merge, or change objects and relations — the same way you could do it by hand. A diagram no longer has to be regenerated from scratch to reflect a change: describing what's different is enough, and the rest of the diagram stays as you left it.

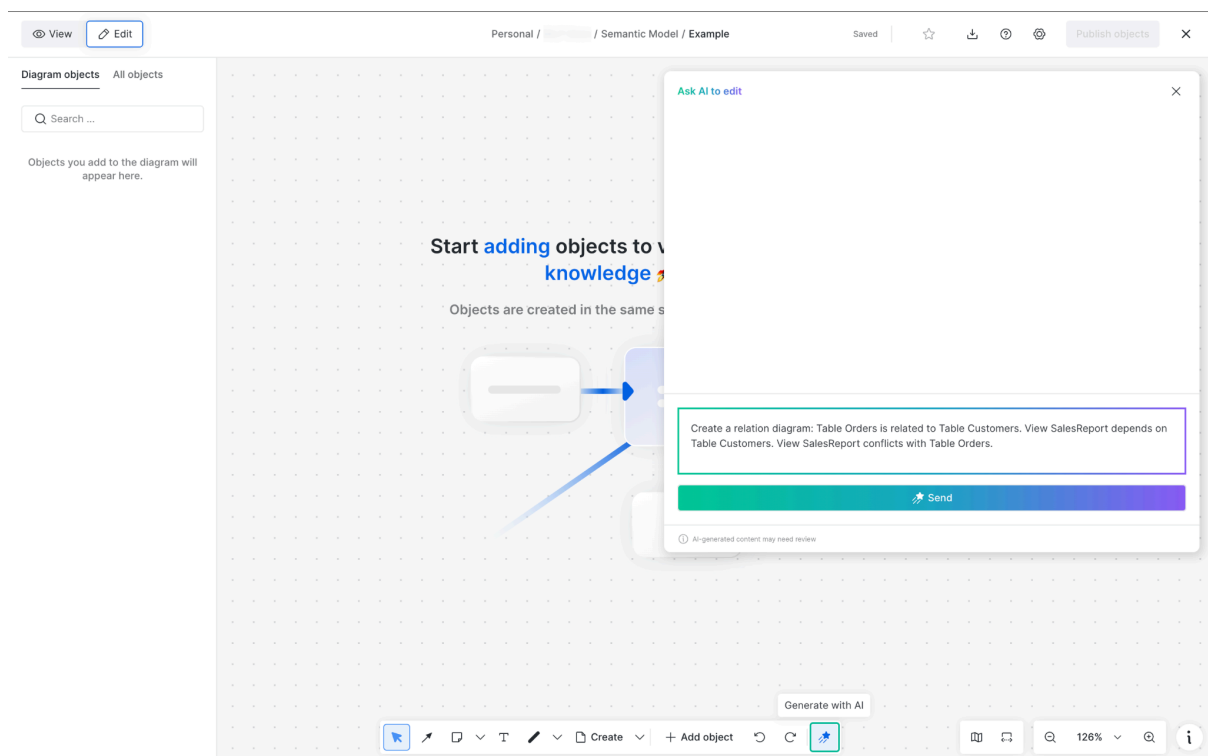


Image 15: The Ask AI to edit panel open over a semantic model diagram, with a prompt describing the relations to create

Drawing & Annotation

Diagram drawing and annotation — notes, shapes, lines, and freehand elements placed directly on the canvas — got smoother and more stable performance. See [Objects and elements](#) and [Actions in diagrams](#) for the full reference.

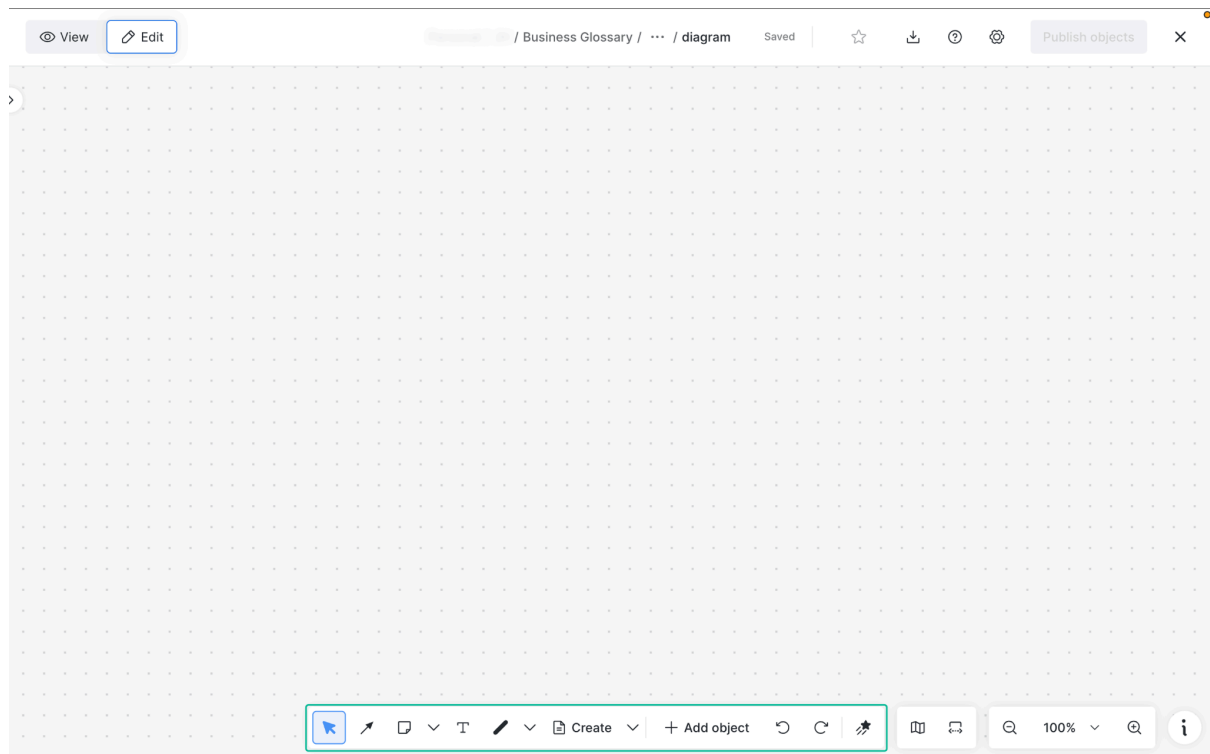


Image 16: A diagram canvas in Edit mode with the bottom drawing toolbar open

Entity Relation Diagrams

Expanded objects in entity relation diagrams now space themselves out automatically. A schema with many tables and columns stays legible as it grows, without manual rearranging.

Knowledge Graph

Knowledge graphs no longer draw every relation of a densely connected object at once. When an object has many relations of the same type, the extra ones collapse into a single grouped node labeled with a count:

- Double-clicking the group opens a paginated, filterable list.
- Selecting objects from it adds them to the canvas individually.
- Once every object in a group has been added, the grouped node disappears.
- Grouping applies at every level of an expansion, not only the diagram's starting object, so even a wide expansion of a heavily interconnected term stays readable.

See [Knowledge graph](#) for the full reference.

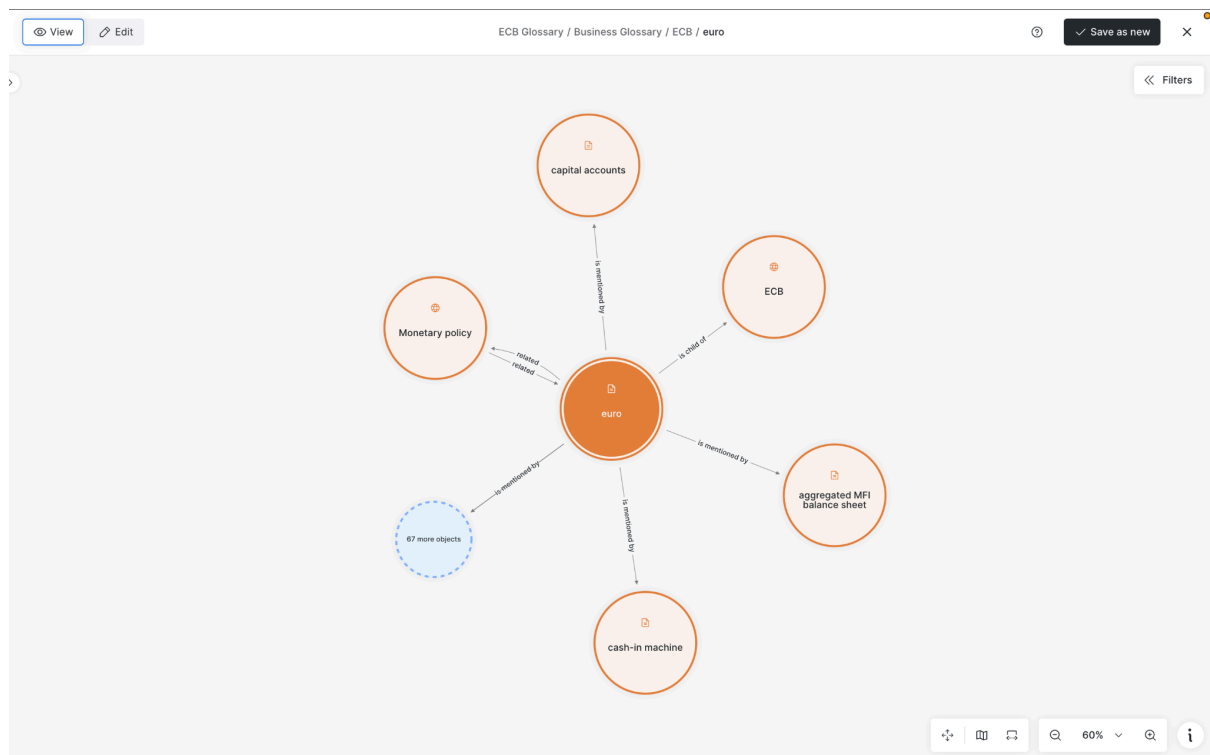


Image 17: A knowledge graph with a dashed grouped node labeled "67 more objects"

Lineage

Lineage diagrams gained two improvements:

- **ELK-based layout:** The layout engine switched to ELK (Eclipse Layout Kernel, an open-source graph-layout library). Large diagrams, on the order of a hundred flows, stay readable instead of collapsing into a tangle, and the layout restores the left-to-right reading direction lineage had before.
- **Power BI column-level tracing:** Relations are now parsed from Power Query M expressions (the formula language behind Power BI's data transformation steps) down to the originating source systems, not just between Power BI objects themselves, and a parsing failure is written to the run log instead of being silently dropped.
- **Custom object-type colors:** An object type can now be assigned its own color, carried into every lineage diagram so a given type is recognizable at a glance across a large flow.

External Link Checking

This feature checks the external links inside your object texts and reports which addresses no longer resolve. A standalone checker performs the checks from your own infrastructure, not from Dawiso itself.

Results appear on a **Broken links** widget that you can add to any dashboard. By default it lists addresses that are currently broken and still referenced somewhere, most recently checked first. Each row carries the object and attribute the link sits in, the address and its result, the check date, and both its referenced and reachable status. See [External link checker](#) for setup and the full field reference.

NAME	ATTRIBUTE	ADDRESS	STATE	CHECKED	REFERENCED	REACHABLE
	Description	https://	404 Not Found		Yes	No
	Summary	https://	404 Not Found		Yes	No
	Description	http://*	DNS failure		Yes	No
	Description	http://*	DNS failure		Yes	No
	Description	http://*	DNS failure		Yes	No
	Description	https://	404 Not Found		Yes	No
	Description	https://	DNS failure		Yes	No
	Description	https://	DNS failure		Yes	No
	Description	https://	DNS failure		Yes	No
	Description	https://	404 Not Found		Yes	No

Image 18: Broken external links widget listing addresses with their state, checked date, referenced, and reachable columns

Import/Export Improvements

Objects can now carry a small data sample directly, and several export paths that previously failed silently got fixed.

Export Reliability

An automatic export that produces an attachment now fails the task outright when the result is empty, rather than reporting success on a zero-byte file as it used to. Any attachment already created that way needs to be regenerated; it won't fix itself.

Additional fixes further improve export reliability:

- Word and ZIP exports now reliably open in Windows Explorer.
- Generating an Excel Template while looking at a historical version of an object now captures that same version, instead of quietly pulling the current one.
- The **Changes tab** again shows word-level differences, restored after a prior regression.

Sample Data

Any object you can edit can now carry a small CSV or JSON data sample, viewable as a sortable, filterable table wherever the object itself is shown. Upload it through the UI, the REST API, or an MCP tool that an AI assistant can also read from when answering. A new upload replaces the old sample outright, and nothing preserves what was there before. Sample values stay out of search results, appearing only in that table. See [Sample data](#) for the full reference.

Interface Redesign

This release introduces a complete redesign of the Dawiso interface, focused on giving your data more room to breathe. The top navigation bar and page header merged into a single, thinner strip, freeing up working space that used to sit under a tall, dark bar. Tables and other components across the interface also became more compact, showing more rows at once without scrolling.

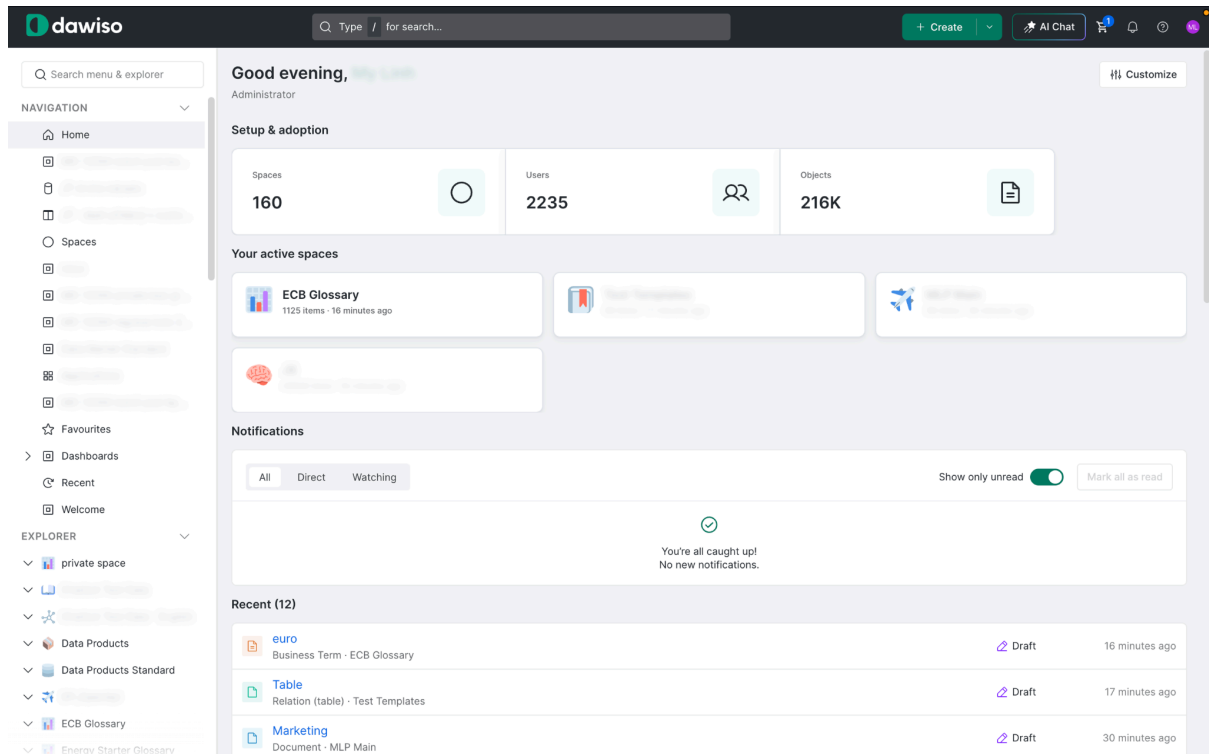
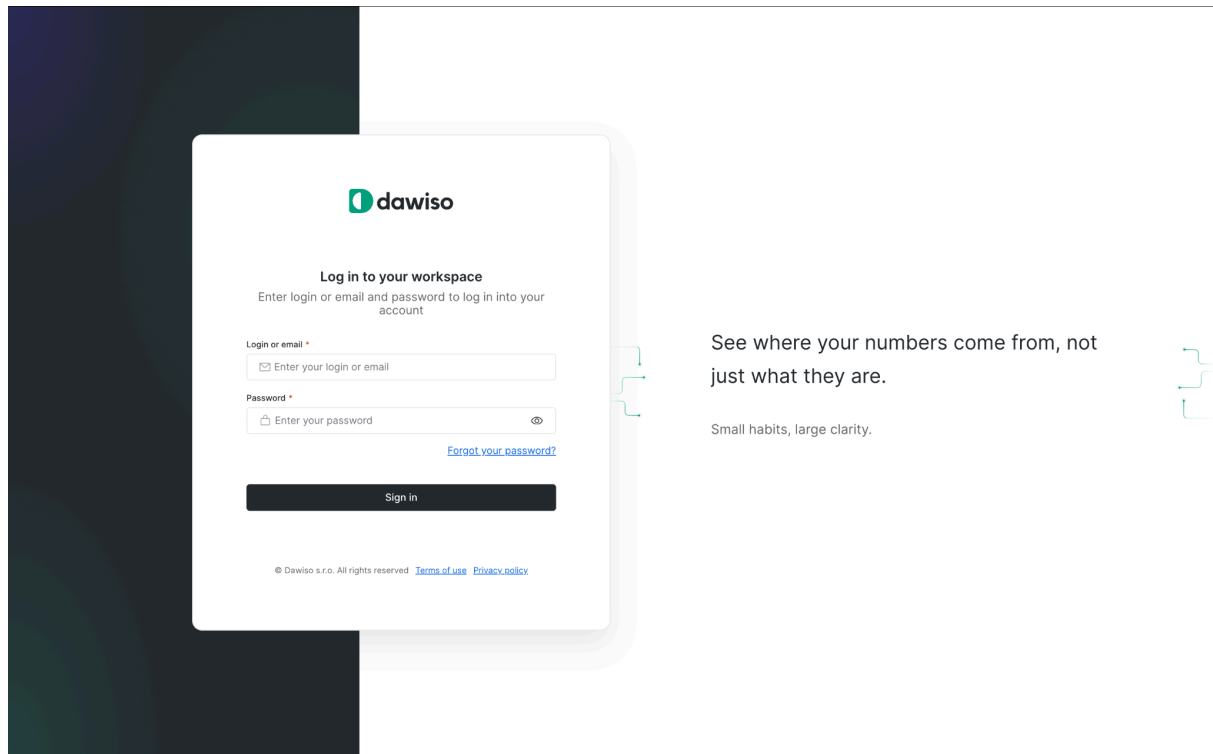


Image 19: The Dawiso dashboard with the merged, thinner top navigation bar

Login Page

The sign-in page was redesigned to match the rest of the interface. A short tagline now rotates below the form every 20 seconds; administrators can replace the default set with their own.



20: The redesigned Dawiso login page, with a rotating tagline and terms-of-use and privacy-policy links

Users & Permissions

The **Users & Permissions** screen moved to a master-detail layout: a list of users and groups on one side, and the details of whichever one is selected on the other, replacing the previous full-page navigation between list and detail views.

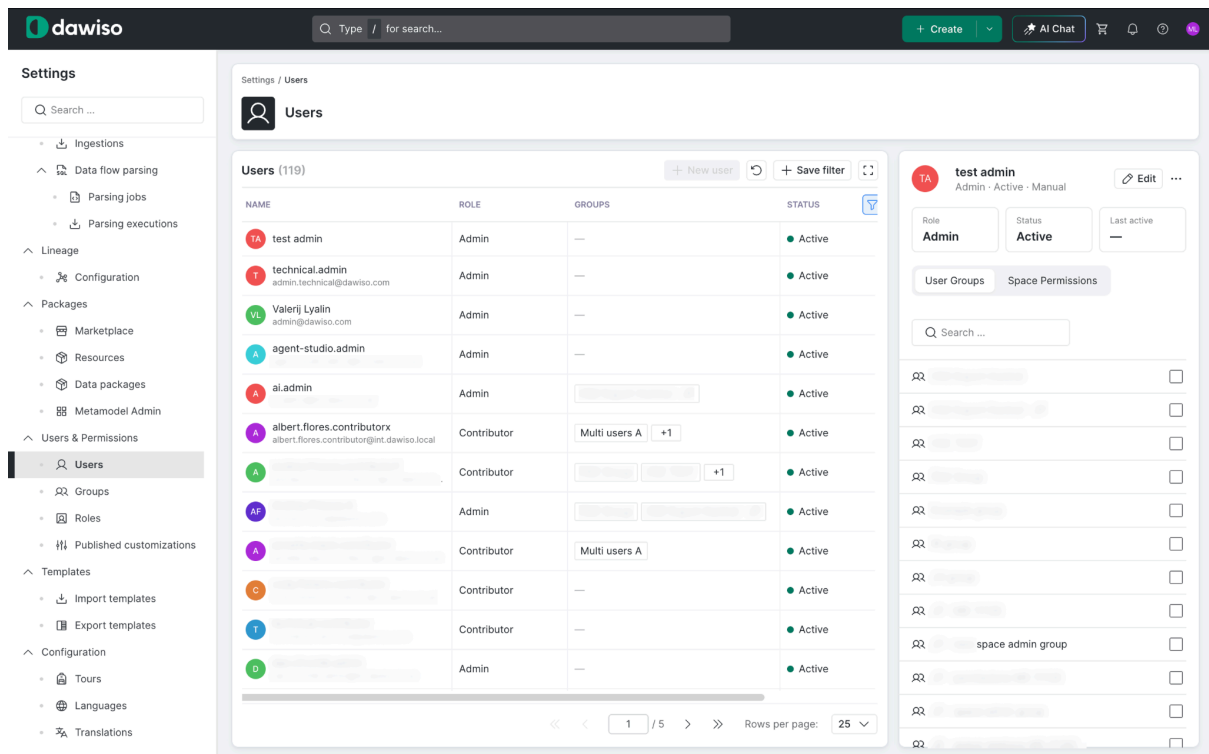


Image 21: The Users screen with a user list on the left and the selected user's details, groups, and permissions on the right

Objects

You can now change an object's type without recreating it, and a one-click shortcut numbers an object's children from zero. Beyond that, moving, renaming, or transitioning an object is far less likely to quietly drop data or in-progress work along the way.

Change Object Type

An object can now change type directly, without recreating it or losing its history. Relations, comments, workflow state, and prior activity all carry over as-is, so reclassifying an object no longer means starting over.

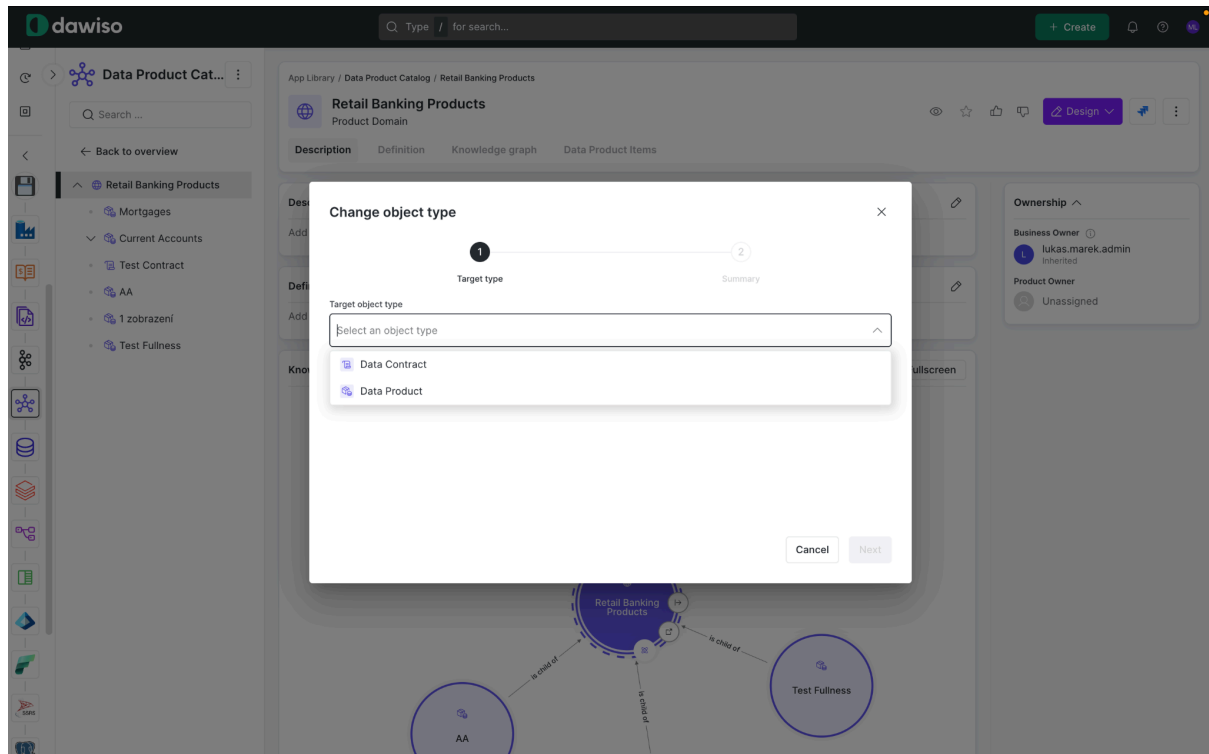


Image 22: Change object type dialog with a Target object type dropdown

Changing type comes with a few rules:

- Attributes the new type doesn't define map to a compatible attribute, or stay stored but hidden.
- A new parent can be set when the new type doesn't allow the current one.
- Only the type, and the parent where applicable, changes — everything else about the object's identity stays intact.
- Switching back to the original type restores any values that were hidden rather than mapped.

See [Object management](#) for the full procedure.

Hierarchy & Inheritance Reliability

Several object and package operations no longer lose data silently. Moving an object to a new parent also moves any automatically created relation between it and its previous parent, such as one configured with `is_creatable_on_child_add` (see [Features \(relation type\)](#)). Previously, that relation was left pointing at the old parent instead.

Additional fixes further improve hierarchy and inheritance reliability:

- Renaming a child object no longer deletes its inherited attributes.
- Deleting a package no longer deletes unrelated system types along with it.
- Creating a child object now respects the object type's "inherit from parent" role setting.
- Cloning an object preserves its inherited roles and attributes.
- A locked transition button's tooltip now explains what's missing (attributes, roles, or relations) before you go looking.
- An unsaved edited attribute on a transition offers **Save**, **Discard**, or **Cancel** instead of losing the change silently.

Object Numbering

Numbering configuration gained a shortcut and now survives more operations without breaking. The **Number direct children from 0** checkbox starts the whole numbering sequence at 0 instead of 1, in one click. A numbering pattern set on an object clears automatically whenever the object is moved, copied, or a parent's numbering is recalculated. It then reverts to inherited numbering. See [Objects](#) for the full numbering reference.

Numbering of all objects in this hierarchy ✕

Numbering starts at level * Numbering pattern

3 0.1.

Numbering starts at this level of the hierarchy; levels above it stay unnumbered. Level 1 numbers the top-level objects.

☒ Number direct children from 0 (deeper levels start at 1)

Example Preview

```

—
  —
    0. numbering starts here
      1.
        1.1.
          2.
  
```

ⓘ This overwrites the numbering settings of all child objects.

Cancel Calculate

Image 23: Numbering dialog with Number direct children from 0 checked

Workflow Transitions

Moving an object to a space with a different workflow now maps its state directly in the move dialog. Only states missing from the target workflow require mapping, and the move can't finish until every listed state has one. See [Object management](#) for the full move procedure.

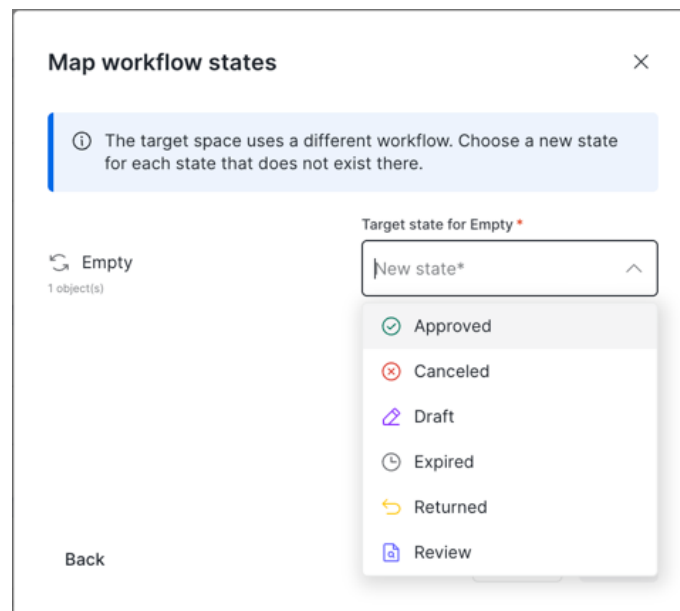


Image 24: Map workflow states dialog with a dropdown of target states

Onboarding & Guided Tours

This release begins a complete rebuild of onboarding: a first-login introduction, a running checklist of first steps, contextual hints on individual interface elements, and guided tours through specific workflows, with each piece configurable by administrators. The new experience ships disabled by default while remaining rough edges are worked out, with a full rollout planned for a later release.

First-Login Walkthrough

The first time someone logs in, a short interactive walkthrough greets them: it visualizes the account's connected apps as a live graph, introduces the AI layer running underneath Dawiso, and lets the new user try asking it something real. The final screen asks what brought them to Dawiso: **Find data**, **Understand data**, **Manage data**, or **Govern data**, each opening a guided tour suited to that choice. **Explore on my own** skips the whole sequence.

Guided Tours

A tour walks through a feature step by step, calling out one screen element at a time; certain steps require actually completing the action shown rather than skipping ahead with a click. Every available tour lives in the **Tours & guides** section of the **Help** menu, each showing its own status (**Completed**, **In progress**, **Ended early**, or **Not started**), and any of them can be replayed from scratch whenever you like. Completing a tour that leads into another offers a **Start tour** button to launch that follow-up immediately.

Administrators can switch guided tours off for the whole environment. See [Guided tours](#) for the full reference.

Onboarding Checklist

Out of the box, a checklist in the top navigation bar tracks progress through ten early tasks (connecting a data source, assigning owners, setting up classifications, and inviting colleagues among them), each one checking itself off the instant you take the matching action anywhere in the app. A task with a linked tour offers a **Show me how** button that starts it directly.

Two panel actions are available: **Minimize**, which keeps the checklist available in a collapsed state, and **Dismiss**, which closes it entirely. A dismissed checklist can be reopened later from the help menu, or restored immediately with the **Undo** option on the confirmation toast. Administrators can turn off parts of onboarding for the whole instance, and chain a guided tour to a specific task. See [Onboarding](#) for the full reference.

Onboarding Hints

An onboarding hint marks a single interface element with a small pulsing dot the first time a user encounters it. Clicking it pops open a short explanation of what that element does, with a **Got it** button to dismiss it and, when a tour is linked, a **Show me how** button to jump straight into it. Hints never block the page, and only one shows per browser tab at a time. Each one is marked seen the moment it appears and never shown to that user again, on any device. The whole feature is disabled by default; turning it on still requires an installed package to supply the actual hint text and targets.

See [Onboarding](#) for the full reference.

Packages

This release introduces several updates to packages and how you can work with them.

Attribute Validation

A package can now restrict an attribute type's values to a regular-expression pattern, using a new `regex_validation` feature paired with `regex_validation_error_translation_key` for the message shown when a value fails it.

A mismatched value is rejected whether it comes from a normal edit, bulk edit, the `Set Attribute` automation action, or an Excel import; an empty value passes unless the attribute is also marked mandatory.

Both features must be defined together, since installing a package with only one of the pair fails at install time, and the pattern itself must compile under the .NET engine's ECMAScript-compatible mode or the same failure applies. See [Attribute Types & Features](#) for the full reference.

Metadata Sync

A package's metadata now syncs automatically from a central file, instead of needing a manual update wherever the package is installed.

New Components

A template component can now also serve as a home screen widget. The `components` asset gained a `widget` property, with `titleKey`, `descriptionKey`, `category`, `defaultWidth/defaultHeight`, `minWidth/minHeight`, and `editableFilters`.

This is the package-authoring mechanism behind homepage and dashboard widgets; the user-facing side of building a homepage or dashboard from widgets is covered under [Customizable Layouts Using Widgets](#). See [components](#) for the full schema reference.

Package Editor

A package version now opens in a full editor for reviewing or changing its JSON definition, name, or description, surfacing its number, key, and current-install status up top:

- **Tabs:** The raw **Code**, any validation **Errors** or **Warnings**, and an **Applied Preview** of the fully resolved definition — for patch packages, this becomes a navigable **Applied Preview Graph** instead of raw JSON.
- **One draft at a time:** Directly changing an already-installed or past version and saving spins up a fresh draft from those edits, and locks every other version to read-only until you either finish or abandon it.
- **Separate save and install actions:** **Save** stores a draft, and **Save & Install** does both, so work in progress can sit without being deployed. You can also save with **Ctrl+S** (or **Cmd+S**).

Every patch links to the package underneath it in the Applied Preview Graph, so a multi-patch chain reads top to bottom as a path ending at the base package. See [Working with packages](#) for the full walkthrough, or [Patch packages — Advanced Guide](#) for the patch-specific reference.

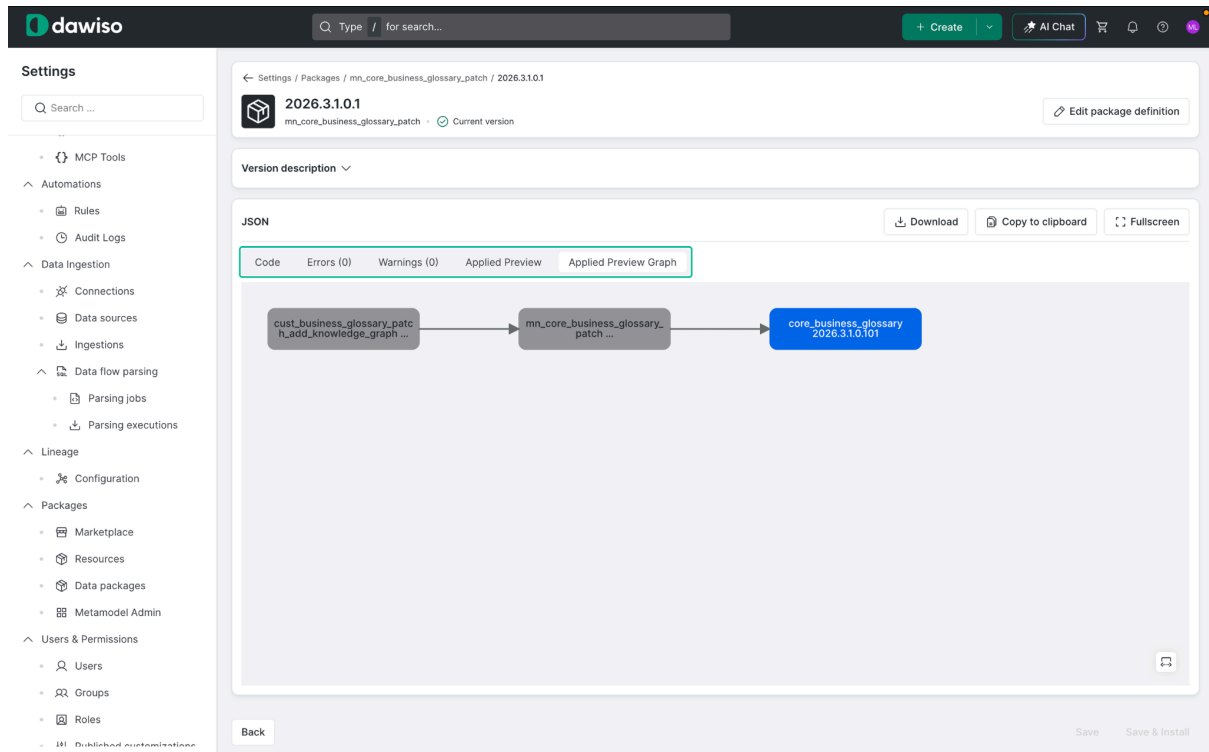


Image 25: A package version's Applied Preview Graph tab, showing a patch chain resolving to the base package

Packages Updates

Downloading every package at once no longer blocks on an immediate file download. Dawiso now assembles the archive out of view and pings you with a notification once it's ready, linking both the file itself and the export task's status page. Leaving the page doesn't interrupt the export.

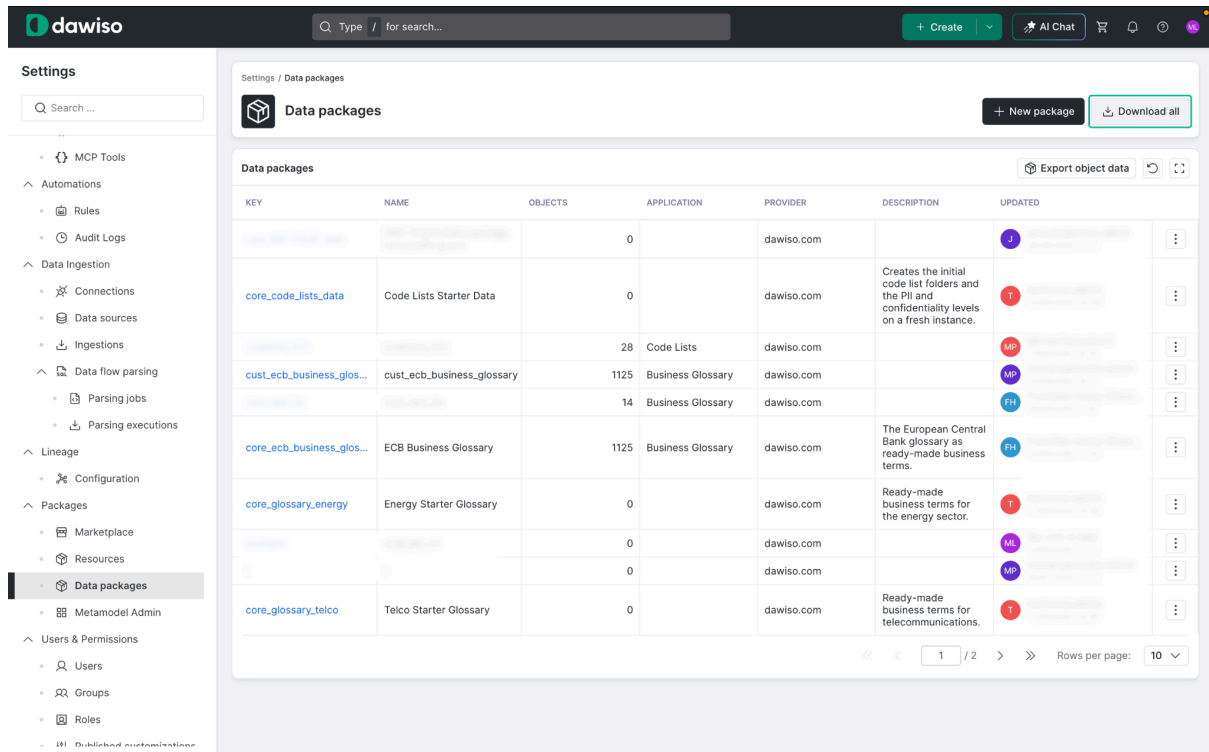


Image 26: A packages list with a Download all button in the top-right corner

Packages: Marketplace Improvements

Enabled by Default

New cloud instances now also ship with the Marketplace enabled by default, so a fresh environment starts with useful defaults already in place rather than a blank slate. Packages flagged for automatic installation set themselves up on their own soon after the environment spins up, with nobody needing to open the Marketplace and click install. See [Marketplace](#) for the full reference.

Reserved core Prefix

The Marketplace now also enforces the **core** package-key prefix as a hard restriction rather than just a best practice: any package key starting with **core**, regardless of case, is blocked at creation, rename, or install time. That prefix is reserved for Dawiso's own default packages. See [Working with packages](#) for the full reference.

Prerequisite Resolution

Installing a package from the Marketplace that depends on others no longer stops to ask for each one separately. Dawiso now works out the full chain of missing prerequisites and installs them together in one step, rather than making an admin resolve and install each dependency by hand before retrying the original package.

Packages: New **data** Type

A new **data** package type moves real content, not configuration, from one Dawiso environment to another.

Automatic Installation

The **autoInstall** property was also clarified: it's valid only on regular and **data** packages, not **patch** or **automation**, and a package only self-installs once every package it **dependsOn** is already installed or is also auto-installing in the same pass. See [package](#) for the full schema reference.

Export Object Data

Export object data, a new admin action in Settings, turns a chosen application's objects in one space directly into package content, no hand-written JSON required. Running the export again on the same application reuses the same object keys, so it only picks up what's new or changed instead of creating duplicates every time. Each exported scope can also be set to only seed its content when the destination is still untouched, so a starter dataset never silently overwrites work someone already did there. After install, the package detail page reports whether the data **seeded**, was **skipped**, or was **partially seeded**, so an admin can tell at a glance whether the package did what it was supposed to. See [Export object data](#) for the full walkthrough.

New Schema

data packages introduce four new schema pieces:

- A new package **type: data**.

- A new [data](#) asset: an array of scopes, each declaring a [space](#) (create one) or [spaceRef](#) (use an existing one), a [dataInstall](#) condition ([always](#), [targetSpaceApplicationEmpty](#), or [targetApplicationEmpty](#)), and the [objects](#), [objectImages](#), and [objectRelations](#) that scope installs.
- A new [objects](#) asset: creates real catalog objects, with [key](#), [objectTypeKey](#), [name](#), [attributes](#), and an optional [parentKey](#) for hierarchy.
- A new [objectImages](#) asset: attaches a binary image to a package-created object, referenced from an attribute value through a [dawiso-pkg://object-image/<imageKey>](#) token.
- A new [objectRelations](#) asset: declares relations between objects the same package creates.

Product Tours and Catalog Objects

Previously, something like a guided tour only worked where it was originally built; moving it elsewhere meant rebuilding every step from scratch, since its internal pointers were tied to that specific environment. A [data](#) package fixes this by matching each piece up by name rather than by internal ID, so the tour lands correctly no matter which environment installs it. Running the same install again later folds updates into the tour that's already there instead of creating a second copy. Two kinds of content ship this way today: product tours, and now whole sets of catalog objects too. See [Data packages](#) for the tour-export walkthrough.

Security & Permissions

Administrators can now restrict hierarchy changes to admin accounts only, tightening who's allowed to restructure the catalog.

A user-relation picker, like the one used to assign an owner or steward, can also be scoped to a specific pool of people. Instead of offering every user or group, it can offer only those who already hold a different relation on the same object, such as limiting reviewer candidates to people already assigned as a contributor. See [codetable-label-user](#) for the full schema reference.

Unified Platform

Dawiso increasingly works as a platform of connected apps rather than one application, and it's also getting faster under the hood. This adds Gate, a new access-management app for data products, a shared identity service connecting Dawiso and Gate together, and a handful of performance fixes across the existing app.

Gate

Gate is a new external app that oversees data products and the output ports they expose, with its own screen for platform mapping, tiers, classification templates, and a nightly sync. Full configuration details for that screen are being documented separately.

Within Dawiso, a **Data Products** panel under Settings lets an admin confirm the Gate integration is healthy: **Test connection** confirms Dawiso can reach Gate, and **Sync with Gate** forces an immediate sync instead of waiting for the regular event-driven one. The panel also shows live status for the two channels Gate authentication uses, checkout requests and catalog sync, and checks that the metamodel has the object, relation, and attribute types the integration depends on, catching a missing type before it causes a silent gap.

A **Last sync run** section reports how the most recent completed sync went: summary counts for what delivered, failed to build, or came through without a classification, plus two expandable lists of problem objects, each row naming the reason and a recommended fix. See [General settings](#) for the full reference.

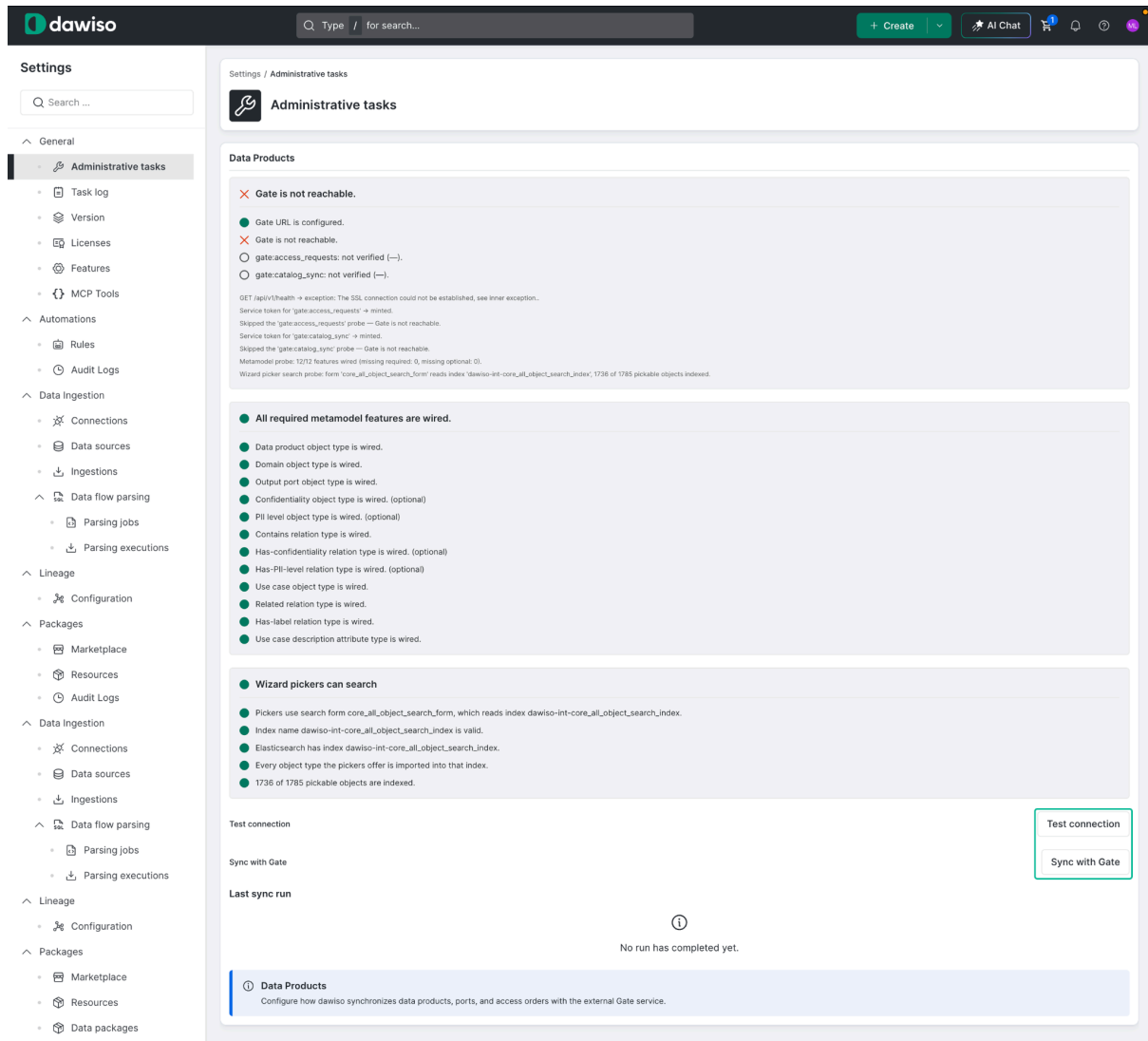


Image 27: Data Products panel with Test connection and Sync with Gate buttons

Identity Service

A shared identity service now sits underneath Dawiso and Gate, so navigation between the two authenticates through one place instead of two separate logins. Dawiso and Gate each connect to it through a single OAuth2 client, and neither side stores a password the other relies on. It's also the service issuing the short-lived tokens the Gate integration status panel checks for. Going forward, it's expected to absorb user loading and other, more involved settings as well.

Platform Performance Improvements

This release brings numerous platform performance improvements, including:

- **Faster app start:** Opening an object's detail page no longer downloads a large, mostly-unused bundle of extra code in the background, and the app also starts faster after login.
- **Faster templates:** Object-type templates now come from a per-version cache instead of being rebuilt every time, cutting a multi-second wait down to a fraction of a second.
- **Faster change-log filtering:** A targeted index replaces a slow full scan, so filtering a large change history stays fast even with hundreds of thousands of records.
- **Fewer redundant redraws:** While someone else is actively editing an object you have open, incoming updates are now batched together instead of forcing a full page redraw for every single one.
- **Large rich text loads safely:** Rich text over 250,000 characters no longer renders the moment you open an object, and the same limit now covers the Activities changelog's version comparison view. A **Show content** button takes over instead, so opening a large document or comparison doesn't freeze the page. See [WYSIWYG HTML editor](#) for the full reference.
- **Operational monitoring:** New health endpoints for Elasticsearch, the database, and the WebSocket hub, plus a Prometheus metrics endpoint covering tasks, ingestion, user sync, and the scheduler, so Zabbix or Grafana can watch Dawiso alongside the rest of your infrastructure.

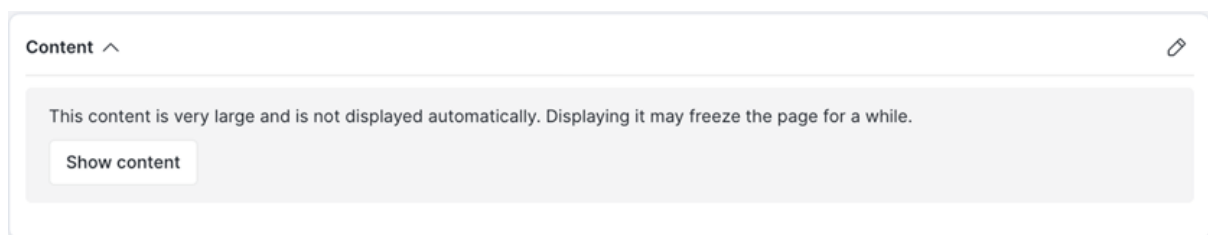


Image 28: A Content panel showing a message that the content is too large to display automatically, with a Show content button

UX/UI Enhancements

We continuously work on improving the user experience of our platform. This release brings a few updates: tooltips shed their sticky behavior, a new in-app satisfaction survey rolls out, and additional fixes landed in navigation, spaces, the rich text editor, the Fullness score, and self-service actions on object pages.

Editor

Images can now be pasted directly from the clipboard, in addition to the existing **Insert Image** button and drag-and-drop. Links no longer inherit unwanted formatting from the surrounding text. Attachments removed before being saved no longer linger in the object's Attachments list afterward. See [WYSIWYG HTML editor](#) for the full reference.

The rich text editor's diagram options also grew beyond Draw.io. Typing **/** now also offers a Lucidchart embed: pasting a Lucidchart link turns it into a live, interactive diagram inside the description, rather than a static image. The Markdown editor gained the same embedded-content capability rich text already had.

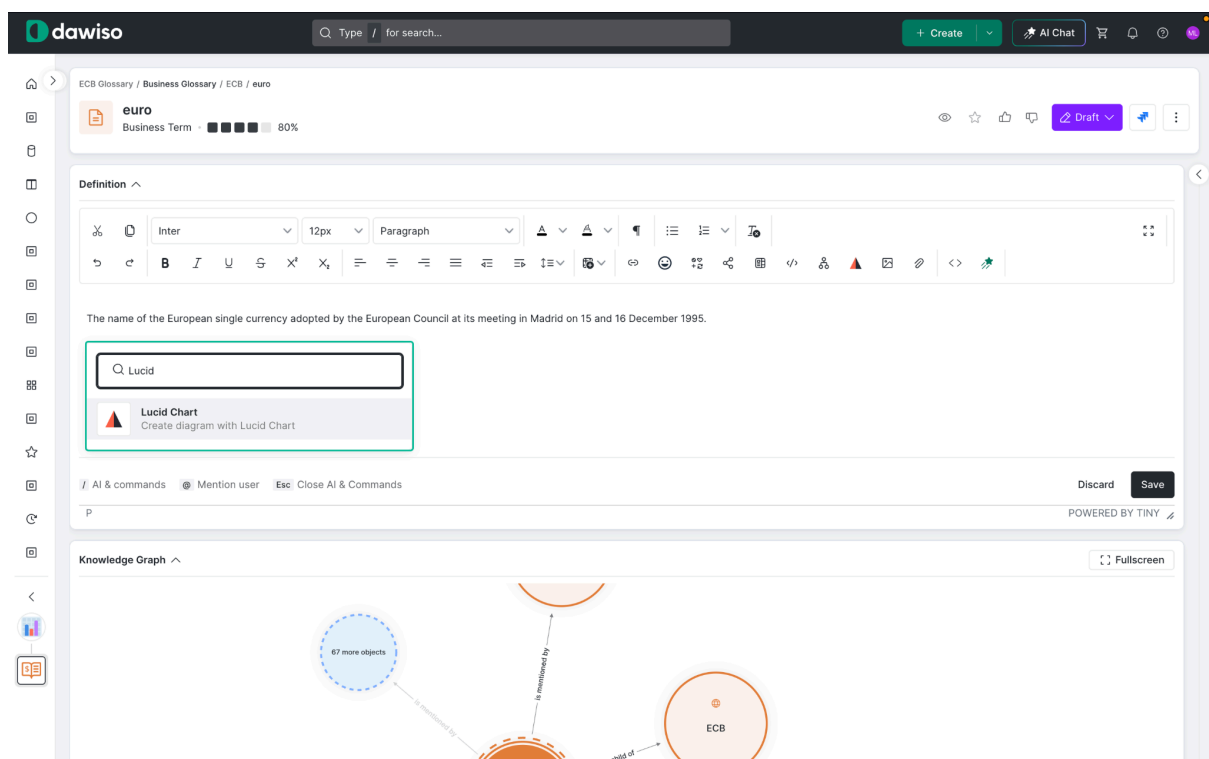


Image 29: The **/** slash menu in the rich text editor, showing the Lucid Chart embed option

Fullness Score Detailed Overview

Hovering over a Fullness score now opens a per-item checklist instead of a single number: it lists each attribute, relation, and user relation the score counts, how much each is worth, and whether it's already been filled in or still blank.

That's enough to spot what's missing without switching over to the separate weight table. The popup is view-only and doesn't total anything itself, so the percentage next to the object stays the one number on screen. See [Business Glossary](#) for the full reference.

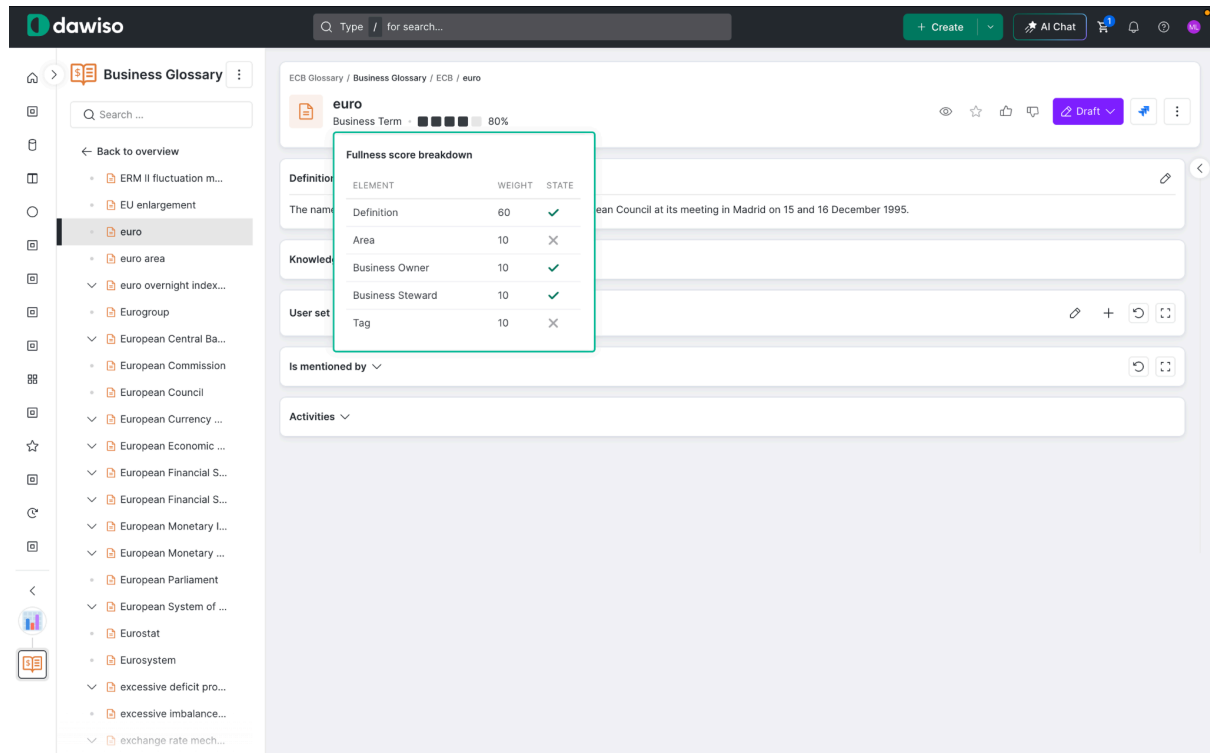


Image 30: Fullness score breakdown popup listing each element with its weight and state

Navigation

The left-side object hierarchy now auto-expands to three levels by default when you open it.

NPS Survey

A short 0–10 satisfaction survey with an open question shows up in the corner of the screen. Server-side logic decides when to trigger it, based on real usage rather than immediately after login, and administrators can turn it off.

Spaces Icons

Creating a space now offers a wider selection of icons to choose from.

Tooltips Improvement

Tooltips across the app are no longer sticky by default, so moving the cursor away closes them right away. Tooltips over content someone might need to copy, such as names, emails, paths, IDs, or error messages, stay open as before.

User Relation Actions

Object pages can now offer self-service buttons for claiming or dropping a user relation, such as acknowledging ownership, instead of always requiring an administrator to assign it. A button's own label can change once it's clicked, and its visibility can be restricted to users who already hold a different, specified relation on the object. The same kind of visibility restriction now also applies to section-title form actions, so a form-opening button can stay hidden from anyone without the right relation. Separately, a user-picker field in forms can allow selecting more than one person at once, instead of only one.