Contents

Admin Role Access Control **1**
1. Overview . . . 1
Key Features . . . 2
Compatibility . . . 2
2. Installation . . . 3
Via Composer . . . 3
Verify . . . 3
3. The Store Scope tab . . . 3
Ticking a website grants its store views . . . 4
Allow access to ALL (unrestricted) . . . 5
The Administrator role is always unrestricted . . . 6
4. How scoping is resolved . . . 7
5. Enforcement . . . 7
Grids and mass actions . . . 8
Single-entity controllers (404 gate) . . . 9
Categories . . . 9
The store switcher and the store context . . . 10
Reports . . . 10
What store scope does not cover . . . 10
6. Attribute Permissions . . . 10
What the restricted admin sees . . . 12
Server-side enforcement . . . 12
7. Configuration . . . 13
8. Command-line health check . . . 14
9. Developer API . . . 14
10. Use cases . . . 14
11. Troubleshooting . . . 15
12. Support & Changelog . . . 15

Admin Role Access Control

User Guide
Version: 1.3.0
Compatibility: Adobe Commerce and Magento Open Source 2.4.7, 2.4.8, 2.4.9 · PHP 8.1 – 8.5
Vendor: Webmaster Ramos

---

1. Overview

Native Magento admin roles control *what* an administrator may do (which menus, grids and actions an ACL role can reach) – but not *where*. On a multi-store installation every back-office user who can open the Orders grid sees orders from **every** website and store view. Role Store Scope adds the missing dimension: it limits each admin role to a chosen set of websites and store views and enforces that boundary across the Orders, Invoices, Shipments, Credit Memos, Customers and Products areas of the admin panel.

The boundary is enforced on three independent layers, so an out-of-scope record cannot be reached even by guessing its URL or id:

- **Grid collections** are filtered to the role's allowed websites / store views.
- **Mass-action collections** (cancel, hold, delete, …) are filtered the same way, so a restricted admin cannot act on an out-of-scope entity in bulk.
- **Single-entity controllers** (view / edit / save / email / print) reject an out-of-scope id with a 404.

Since version 1.2.0 the module adds a second dimension – *what fields* an administrator may see. **Attribute Permissions** hide chosen product and category attributes and attribute groups from a role, both in the admin edit form and, for products, on save – so a restricted administrator cannot view or change a field they are not allowed to (for example, hiding Cost or margin fields from a content editor).

The module is rule-based and deterministic. It calls no external service and performs no profiling – access is resolved purely from the websites, store views and attributes you assign to each role.

Key Features

- **Attribute Permissions (1.2.0)** – hide product and category attributes and attribute groups from a role, in **Deny** mode (hide the selected) or **Allow** mode (show only the selected), per entity.
- **Server-side enforcement** of attribute permissions – hidden product attributes are removed from the save request, so they cannot be changed by a crafted request; hiding a required field does not break saving.
- A hierarchical **Store Scope** tab on each admin role, rendering every website with its store views as a checkbox tree.
- **Zero-trust default** – a freshly created role is restricted until an operator explicitly grants access (the Magento administrator role is always unrestricted).
- Ticking a website automatically grants all of its store views.
- An **Allow access to ALL** toggle to opt a role out of scoping entirely.
- Enforcement across Orders, Invoices, Shipments, Credit Memos, Customers and Products – grids, mass actions and single-entity controllers.
- A **configurable out-of-scope message** shown to restricted admins.
- A **health-check command** (`webramos:role-permissions:status`) reporting each role's scope.
- A read API (`RoleScopeProvider`) for other modules that must honour the same scope.
- Foreign keys with `ON DELETE CASCADE` – scope rows are removed automatically when a role is deleted.

Compatibility

Platform	Versions
Adobe Commerce	2.4.7, 2.4.8, 2.4.9
Magento Open Source	2.4.7, 2.4.8, 2.4.9
PHP	8.1, 8.2, 8.3, 8.4, 8.5
Scope	Admin panel only – no storefront footprint, theme-agnostic

---

2. Installation

Via Composer

```
composer require webmaster-ramos/module-role-permissions
bin/magento module:enable WebRamos_RolePermissions
bin/magento setup:upgrade
bin/magento setup:di:compile     # production mode only
bin/magento cache:flush
```

Verify

```
bin/magento module:status WebRamos_RolePermissions
```

`setup:upgrade` creates the module's tables via declarative schema (`webramos_role_permissions_website` / `_storeview` / `_settings` for store scope, and `_attribute` / `_attribute_group` / `_attribute_mode` for attribute permissions). No core tables are modified.

---

3. The Store Scope tab

Go to **System → Permissions → User Roles**, open (or create) a role, then select the **Store Scope** tab.

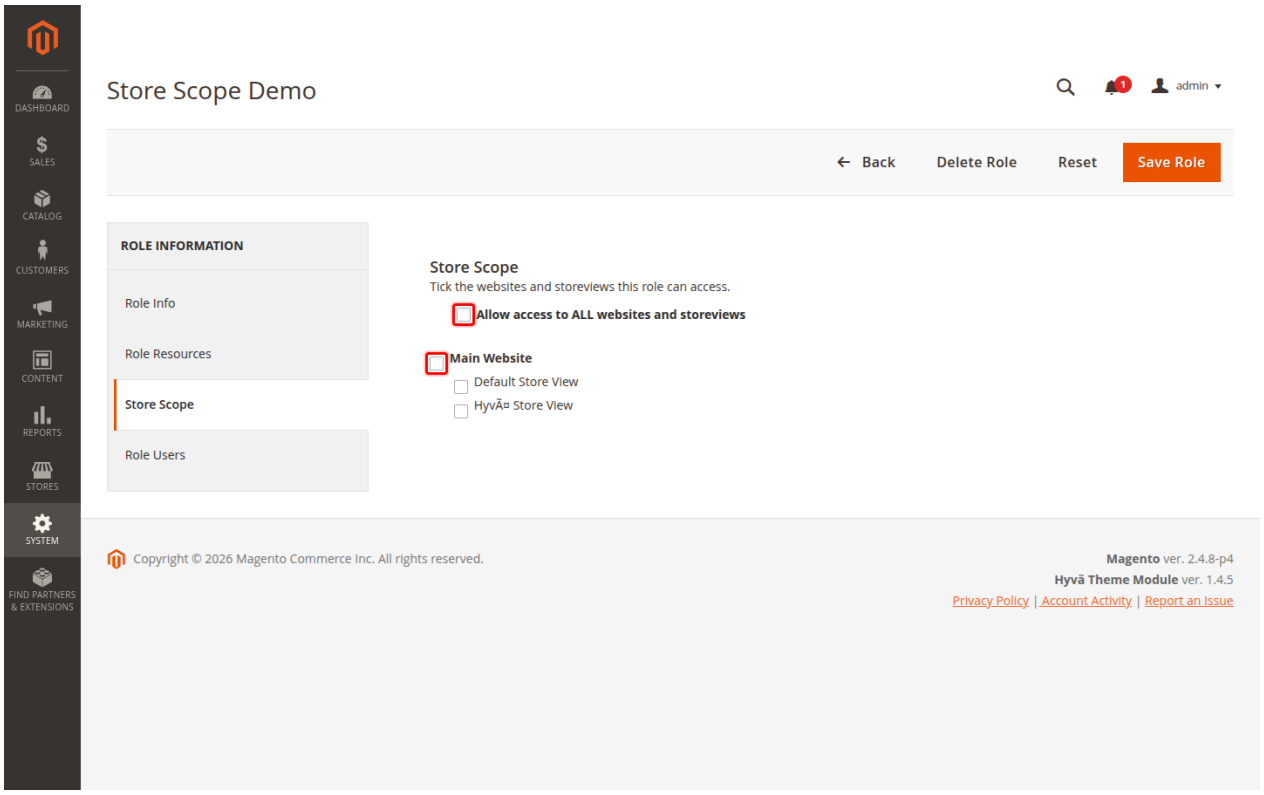

Figure 1: Store Scope tab – zero-trust default

The tab renders every website with its store views as a checkbox tree, plus an **Allow access to ALL websites and store views** toggle.

Control	Effect
Website checkbox	Grants the role access to that website and all of its store views.
Store-view checkbox	Grants access to that individual store view.
Allow access to ALL ...	Marks the role unrestricted – it sees every website and store view (the tree is disabled while this is on).

Editing a role's scope requires the **Edit Role Store Scope** (`WebRamos_RolePermissions::role_scope`) ACL resource, nested under *System* → *Permissions*.

> **Saving:** the role's scope is written when you click **Save Role**. Set the state you want and save – ticking a website, ticking store views, or turning on *Allow access to ALL*.

Ticking a website grants its store views

Ticking a website automatically ticks all of its store views.

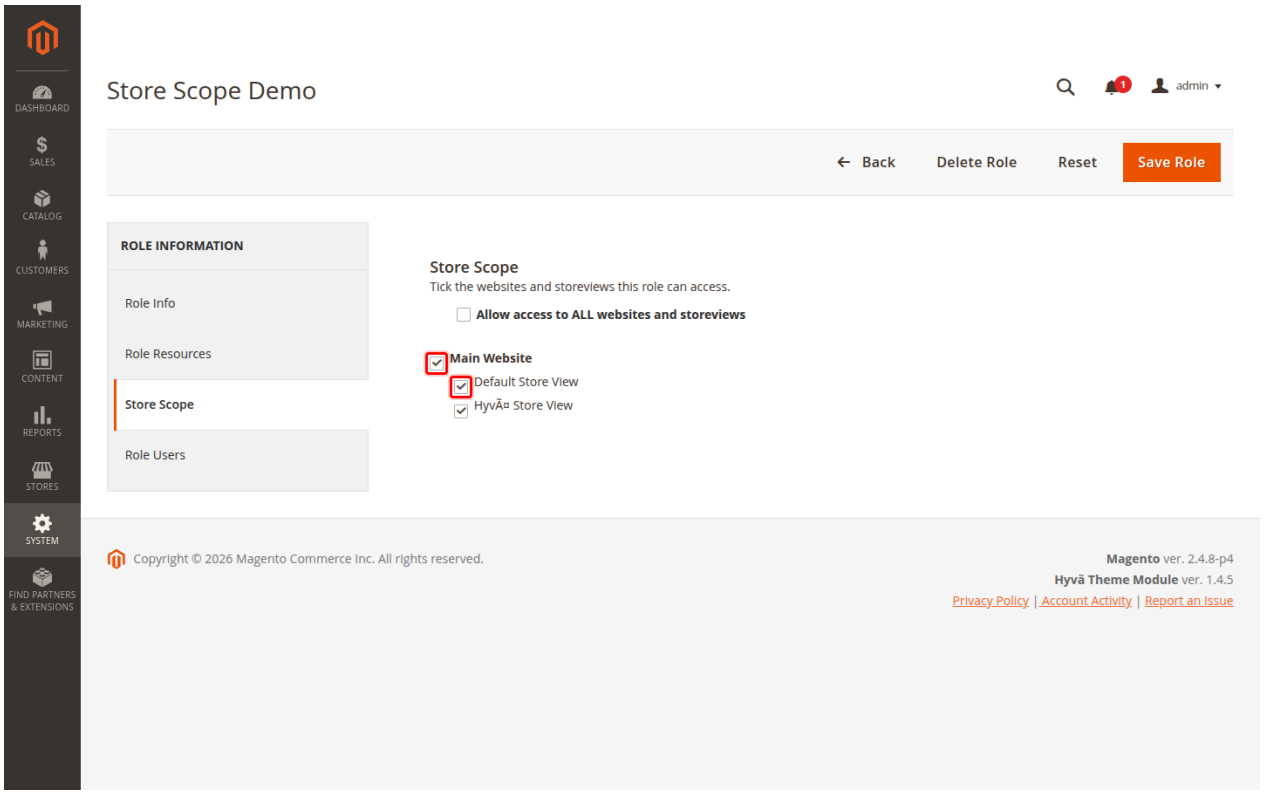

Figure 2: Website ticked – store views auto-selected

Allow access to ALL (unrestricted)

Turning on **Allow access to ALL websites and store views** opts the role out of scoping. The tree is disabled to show the role is unrestricted.

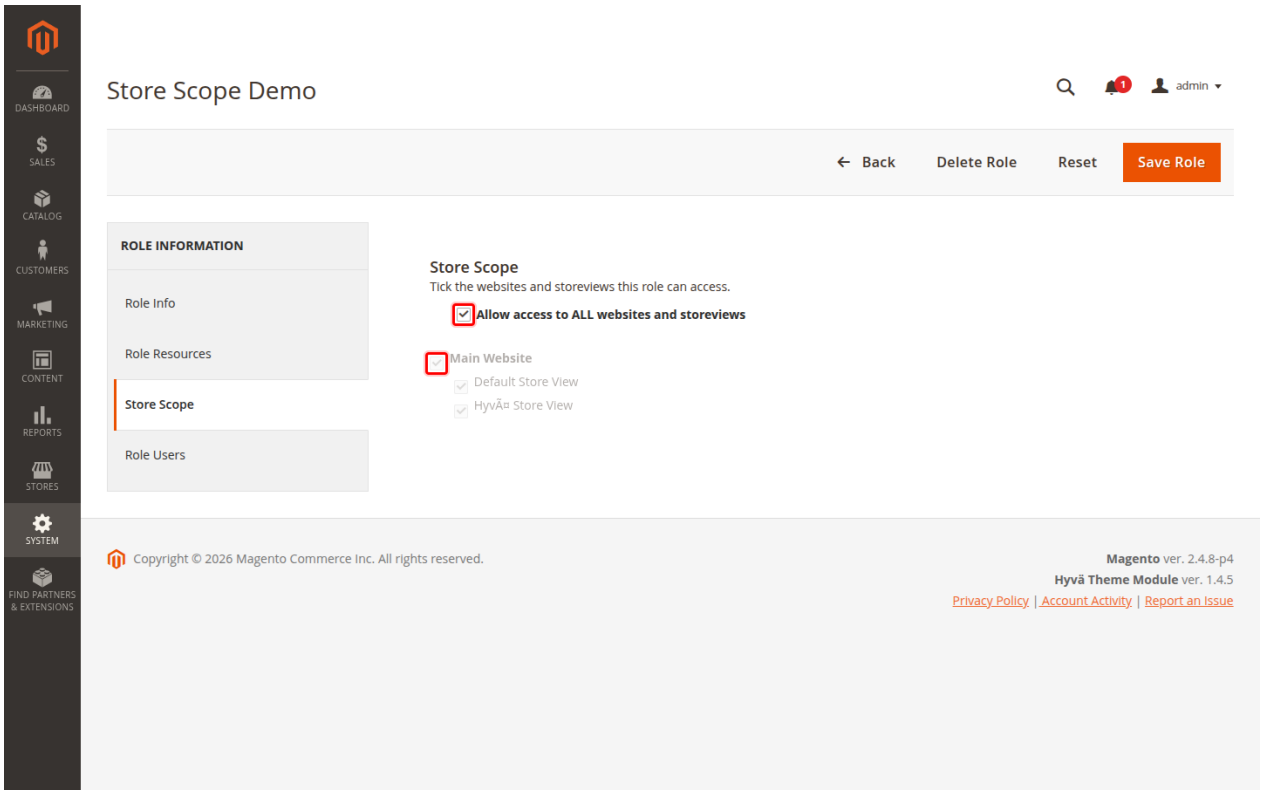

Figure 3: Allow access to ALL toggle

The Administrator role is always unrestricted

The built-in Magento Administrator role (`role_id = 1`) is always unrestricted; its toggle is checked and locked.

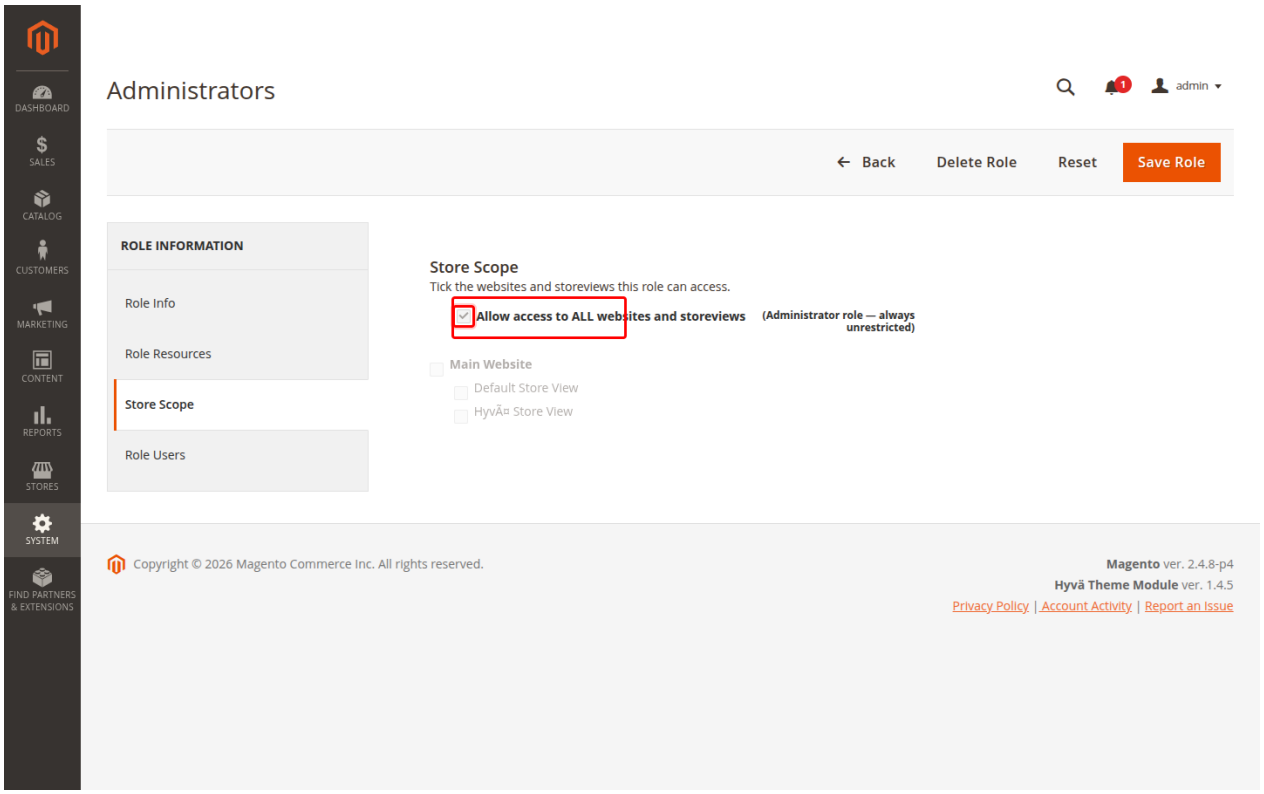

Figure 4: Administrator role locked

---

4. How scoping is resolved

A role is **unrestricted** (sees everything) when either holds:

- it is the Administrator role (`role_id = 1`); or
- it has the **Allow access to ALL** toggle on (an explicit `is_unrestricted = 1` settings row).

Otherwise the role is **restricted** and may access only:

- the websites you ticked, and
- the store views you ticked, **plus** every store view of a ticked website.

Zero-trust default: a freshly created role that has never been granted any website or store view, and does not have *Allow access to ALL* on, is restricted to nothing – its grids are empty and direct URLs to any entity return a 404. Grant it a website (or turn on *Allow access to ALL*) to open access.

---

5. Enforcement

Once a role is restricted, the boundary applies everywhere a store-scoped entity is listed or opened.

Grids and mass actions

The Orders, Invoices, Shipments, Credit Memos, Customers and Products grids are filtered to the role's allowed websites / store views. A restricted admin whose scope does not include an order's store sees an empty grid.

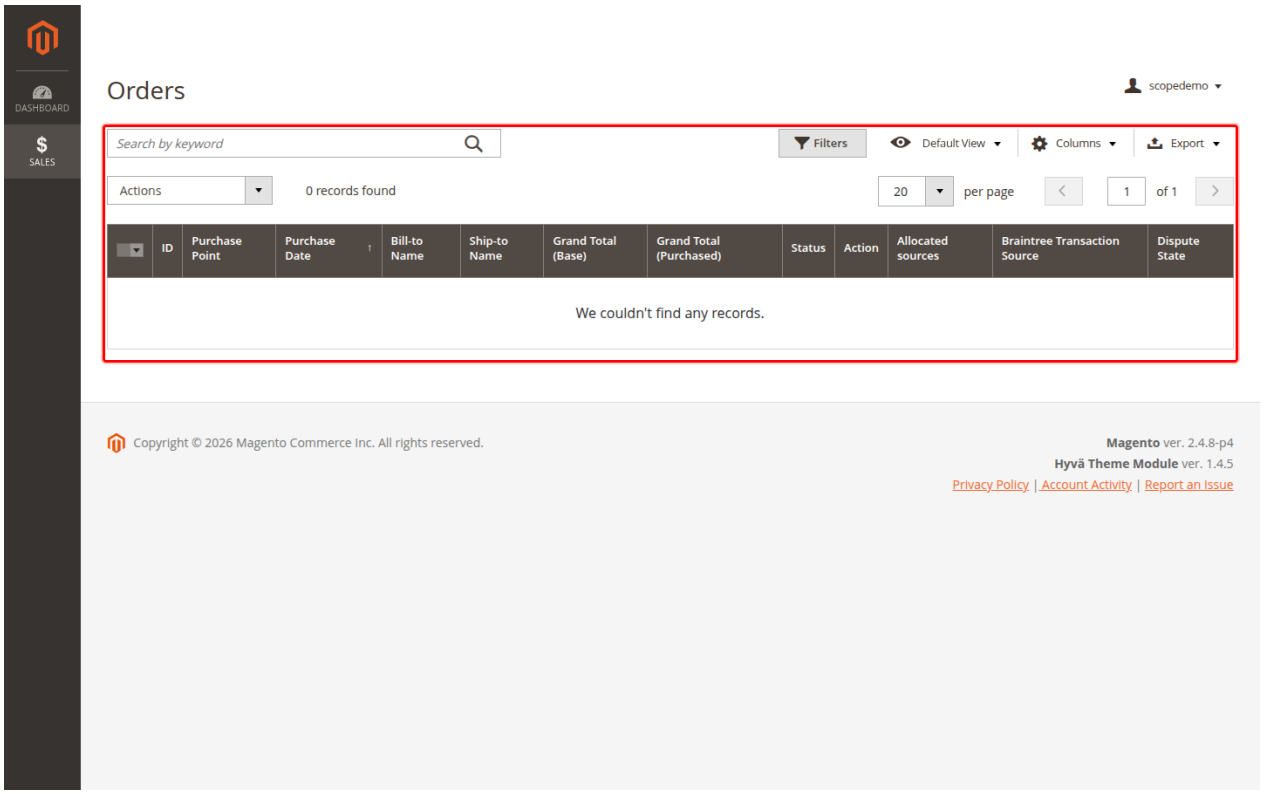

Figure 5: Restricted admin – Orders grid empty (out of scope)

After the order's store is granted to the role, the same grid lists it.

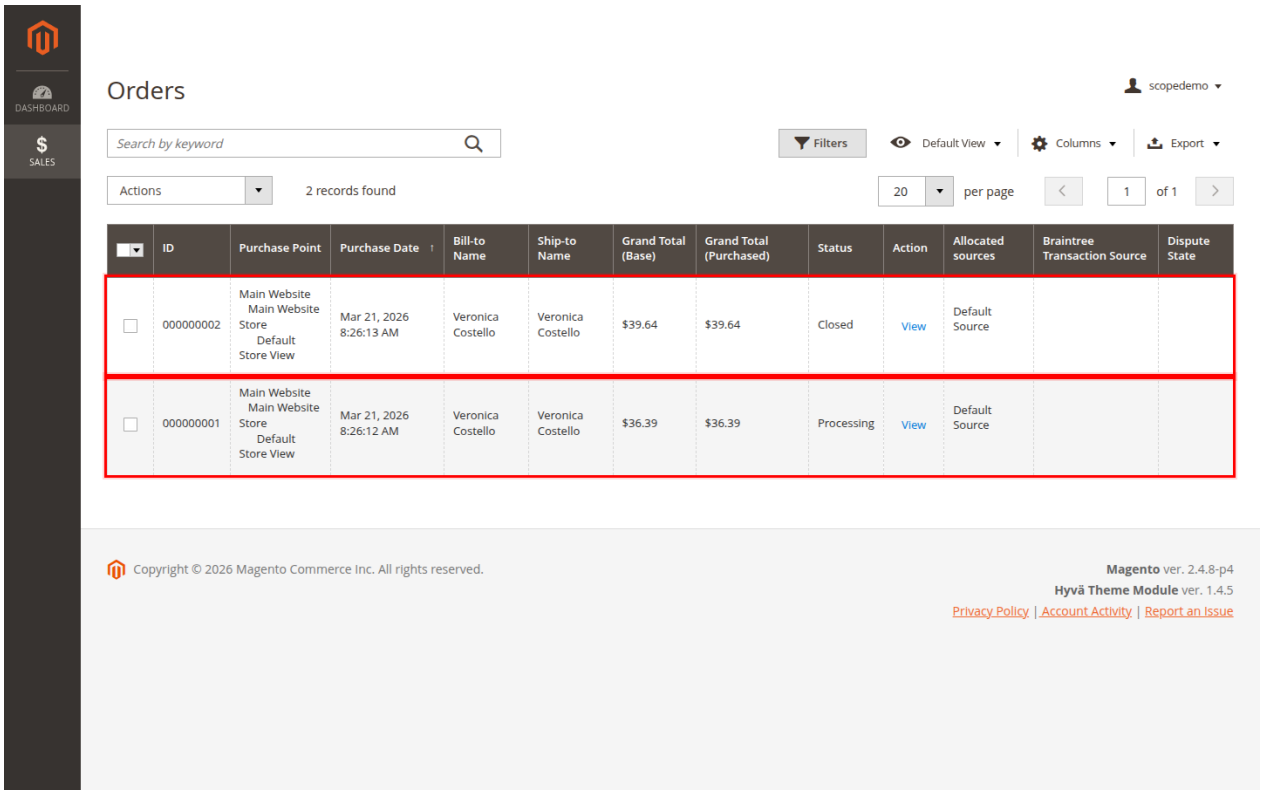

Figure 6: Restricted admin – Orders visible after grant

Mass actions use the same filter, so a restricted admin cannot cancel, hold or delete an out-of-scope record in bulk.

Single-entity controllers (404 gate)

Even with a direct URL, opening, editing, saving, emailing or printing an out-of-scope Order, Invoice, Shipment, Credit Memo, Customer or Product is rejected with a 404 by `ScopeEnforcer` – the grid filter and the controller gate back each other up.

Categories

Categories have no store column in Magento, so their scope is derived: each store view belongs to a store group, and each store group has a root category. A restricted role may work with the root categories of its granted store views and everything beneath them.

That scope applies to every admin surface that lists categories – the Catalog → Categories tree, the category picker on the product form, the new-category dialog, category autocomplete, the widget chooser and admin global search. Categories above a granted root remain visible, because the tree cannot render without them, but they are never editable.

Writing is gated the same way: creating, saving, moving or deleting a category outside the role's scope is refused with a 404, and so is moving or creating a category *into* one. Opening Catalog → Categories without picking a category lands a restricted admin on their own granted root rather than the default store's.

The store switcher and the store context

Every admin screen with a store switcher offers only the store views the role was granted. A website or store group whose store views are all out of scope disappears from the control – there is nothing to click that would be refused.

Behind the control, the store a request is addressed to is validated before the page runs, on categories, products, mass attribute updates, custom variables, system configuration and admin order creation. A store view the role was never granted is refused even if the parameter is supplied by hand.

"All Store Views" – the default scope – is always available, to every role. It is what ordinary admin forms submit, and denying it would break them; see *What store scope does not cover* below for what that implies.

Reports

The report screens that carry a store switcher report on granted store views only. Leaving the store filter empty means *all stores* in stock Magento; for a restricted role it now means *all granted stores*. The dashboard's Bestsellers tab is scoped the same way.

What store scope does not cover

Three behaviours follow from Magento's own scope model rather than from this extension, and no setting here changes them.

Values saved at a broader scope propagate. A value saved at "All Store Views", or at a website or store group that contains a granted store view, reaches store views the role was not granted – unless those store views override it. This is Magento's scope fallback, the same mechanism that lets you set a value once for a whole website. Where a role must be fenced out of a store view's values entirely, make sure that store view overrides them.

Two report screens have no store-scoped data. Low Stock lists catalogue-wide stock levels, and Downloads sums download counts across all stores – neither holds per-store rows to filter. Their store switcher is filtered like every other; their figures are not store-specific to begin with.

Asynchronous configuration saving bypasses the store check. If your deployment enables Magento's `config/async` flag (off by default), a system configuration save is queued for a background process before the store check runs. Every other screen is unaffected, and the switcher on the configuration page is still filtered. If you rely on store scope for configuration writes, leave that flag off.

---

6. Attribute Permissions

Attribute Permissions let you hide individual product and category attributes, or whole attribute groups, from an admin role. They are configured on the same **User Roles** page as Store Scope, on the **Attribute Permissions** tab.

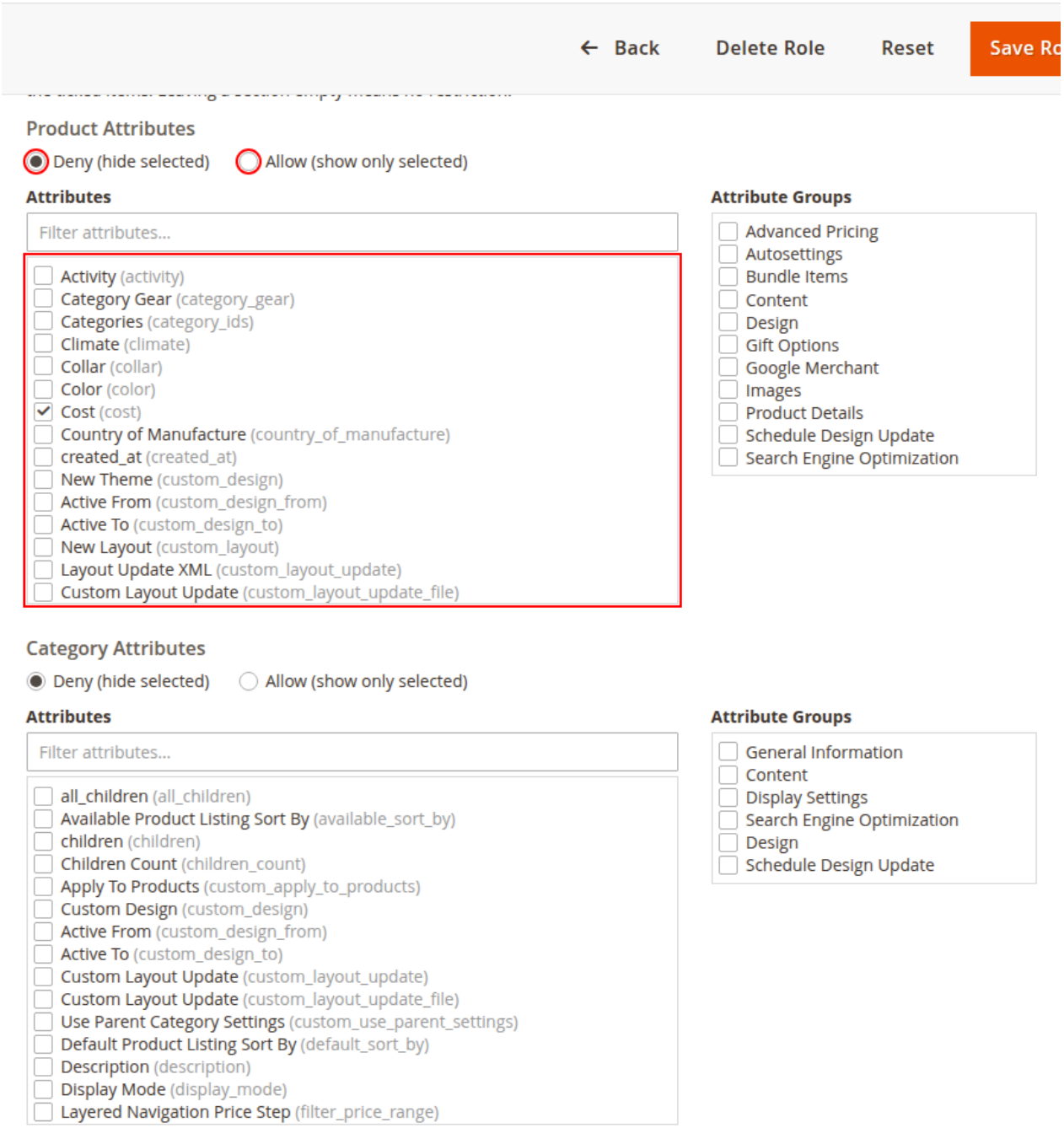

Figure 7: Attribute Permissions tab

The tab has one section per entity – **Product Attributes** and **Category Attributes** – each with a searchable list of attributes and a list of attribute groups, plus a mode toggle:

- **Deny (hide selected)** – the ticked attributes and groups are hidden; every other field stays visible. Use this to hide a few sensitive fields, e.g. Cost or margin fields from a content editor.

- **Allow (show only selected)** – only the ticked attributes and groups are visible; everything else is hidden. Use this to lock a role down to a small set of fields.

Leaving a section empty means the role has no attribute restriction for that entity. The **Administrator** role always sees every field.

What the restricted admin sees

Hidden attributes and emptied groups disappear from the product and category edit forms for that role.

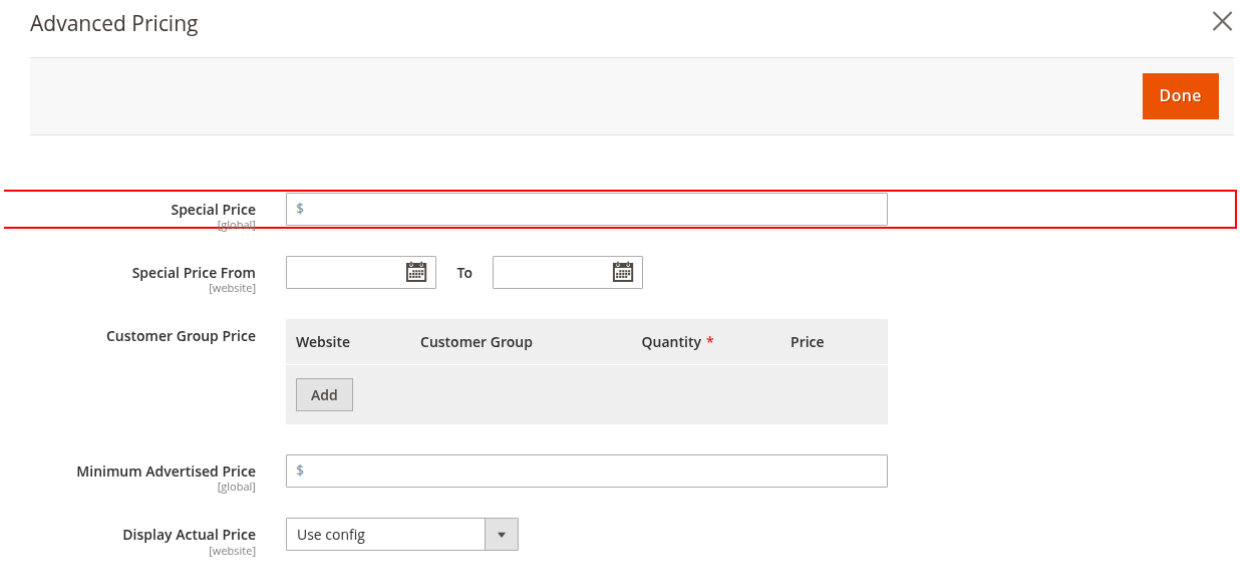

Figure 8: Product form – Cost hidden for the restricted role

Server-side enforcement

Hiding a field in the form is only half the protection. For products, hidden attributes are also removed from the save request, so a restricted admin cannot change a hidden attribute even with a hand-crafted request. Because the value is simply omitted (not blanked), the product keeps its stored value – so hiding a *required* attribute does not prevent the admin from saving the product.

> **Attribute-permission scope.** Product enforcement covers attributes and groups, in both Deny and Allow modes, in the form and on save. Category attributes are hidden

in the form in both modes and enforced on save for the attribute (Deny) case; category group and Allow-mode save enforcement, and hiding attribute columns on the product grid, are planned for a later release.

7. Configuration

Navigate to **Stores** → **Configuration** → **Webmaster Ramos** → **Role Permissions** → **General**.

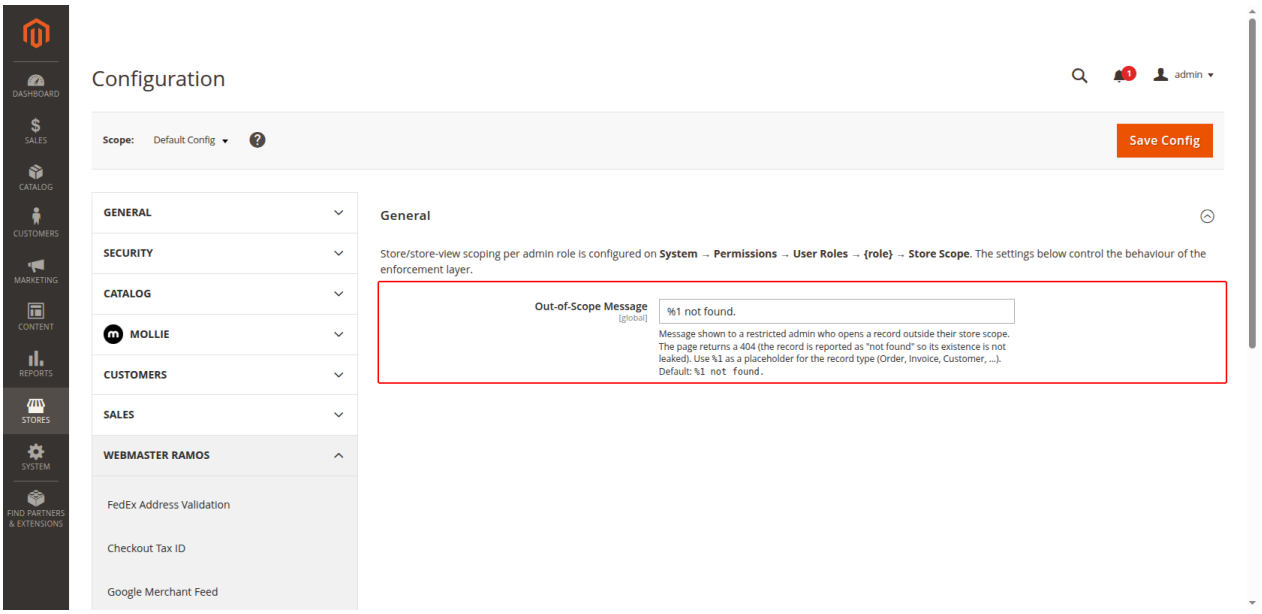

Figure 9: Configuration – General

Field	Description	Default
Out-of-Scope Message	The message shown to a restricted admin who opens a record outside their scope. The page returns a 404; use `%1` as a placeholder for the record type (Order, Invoice, Customer, …). The wording stays “not found” so the record’s existence is not revealed.	`%1 not found.`

Editing this section requires the **Role Permissions** (`WebRamos_RolePermissions::config`) ACL resource under *Stores* → *Settings* → *Configuration*.

8. Command-line health check

The module ships a read-only console command that reports the current state of store-scope enforcement:

```
bin/magento webramos:role-permissions:status
```

It lists every admin role as unrestricted or restricted (with its website and store-view counts), prints the active out-of-scope message, and returns a non-zero exit code if the scope tables cannot be read – suitable as a cron / CI / monitoring probe.

> **Security model.** Store-scope enforcement is active only while the module is enabled. Disabling or removing the module restores full multi-store access to every restricted role (the same behaviour as Adobe Commerce's own store-scope feature, which is also delivered as a module). The command is registered by the module, so it cannot report its own absence – to detect that the module was disabled, monitor it from outside with `bin/magento module:status WebRamos_RolePermissions`.

---

9. Developer API

Other modules can honour the same scope through the read service `WebRamos\RolePermissions\Service\RoleScopeProvider`:

```php
use WebRamos\RolePermissions\Service\RoleScopeProvider;

public function __construct(private readonly RoleScopeProvider $scope) {}

$this->scope->isUnrestricted($roleId);          // bool
$this->scope->getAllowedWebsiteIds($roleId);    // int[] directly assigned websites
$this->scope->getAllowedStoreIds($roleId);      // int[] union: direct + website-derived
$this->scope->hasWebsiteAccess($roleId, $websiteId);
$this->scope->hasStoreAccess($roleId, $storeId);
```

Results are cached per request.

---

10. Use cases

- **Regional teams.** A retailer running one website per country gives each country's back-office team a role scoped to that website. Each team works only its own orders and customers; nobody browses another region's data.
- **Outsourced support.** A support agency role is scoped to a single store view. Agents handle that store's orders and credit memos but cannot reach the rest of the catalogue or customer base.
- **Staging / secondary brand isolation.** A role scoped to a secondary brand's website keeps that brand's staff out of the flagship store's data.

---

11. Troubleshooting

Symptom	Cause / resolution
A restricted admin sees an empty grid everywhere	The role has no scope rows (zero-trust default). Grant a website / store view, or turn on *Allow access to ALL*.
A direct entity URL returns 404 for a restricted admin	Expected – that entity is outside the role's scope. Grant the relevant store.
Scope change does not take effect	Scope is read live; no cache flush is needed. Confirm you clicked **Save Role** and that the role is not the Administrator role (always unrestricted).
The Store Scope tab is missing	Confirm the module is enabled (`module:status`) and the admin user's role has the *Edit Role Store Scope* ACL resource.
Restricted roles suddenly see every store again	Enforcement runs only while the module is enabled. Check `bin/magento module:status WebRamos_RolePermissions` – if it was disabled or removed (e.g. during an upgrade), re-enable it. See §7 *Command-line health check*.
A restricted admin's report totals are lower than an unrestricted admin's	Expected – reports are scoped to the role's granted store views (§5 *Reports*).
The category tree looks empty for a restricted admin	The role's granted store views resolve to no root category, or to a root that has no categories yet. Check which store views are granted and which store group each belongs to.
A store view is missing from the store switcher	Expected – the switcher offers granted store views only (§5 *The store switcher and the store context*). Grant it on the Store Scope tab.
A configuration value saved at "All Store Views" shows up in a store view the role cannot access	Expected – Magento's scope fallback (§5 *What store scope does not cover*). Override the value at that store view.

Logs: standard Magento `var/log/system.log` and `var/log/exception.log`.

---

12. Support & Changelog

- **Vendor:** Webmaster Ramos
- **Email:** contact@webmaster-ramos.com
- **Website:** https://webmaster-ramos.com
- **Changelog:** see `CHANGELOG.md` inside the package.