REST API

Custom Catalog Documentation

Guide for Magento administrators, technical project owners, and implementation teams.

LAST UPDATED
2026-10-13

SOURCE
Open this article online

EXTENSION
Magento 2 Extension Custom Catalog

REST API

The optional package `mageb2b/custom-catalog-api` exposes named catalog management and assignment services through Magento Web API.

```
composer require mageb2b/custom-catalog-api:*
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Catalog records

Method	Endpoint	ACL resource
`GET`	`/V1/customcatalog/catalog/:id`	`MageB2B_CustomCatalog::manage`
`GET`	`/V1/customcatalog/catalog/search`	`MageB2B_CustomCatalog::manage`
`POST`	`/V1/customcatalog/catalog`	`MageB2B_CustomCatalog::save`
`PUT`	`/V1/customcatalog/catalog/:id`	`MageB2B_CustomCatalog::save`
`DELETE`	`/V1/customcatalog/catalog/:id`	`MageB2B_CustomCatalog::delete`

`DELETE` returns `true` when the catalog was removed.

Assignments

Catalog-centric routes assign and unassign products, customers and customer groups below `/V1/customcatalog/catalog/:catalogId/...`:

Method	Endpoint	Body parameters	ACL resource
`POST`	`/V1/customcatalog/catalog/:catalogId/products/assign`	`productIds`	`MageB2B_CustomCatalog::save`
`POST`	`/V1/customcatalog/catalog/:catalogId/products/unassign`	`productIds`	`MageB2B_CustomCatalog::delete`
`POST`	`/V1/customcatalog/catalog/:catalogId/customers/assign`	`customerIds`, `storeId`	`MageB2B_CustomCatalog::save`
`POST`	`/V1/customcatalog/catalog/:catalogId/customers/unassign`	`customerIds`, `storeId`	`MageB2B_CustomCatalog::delete`
`POST`	`/V1/customcatalog/catalog/:catalogId/customer-groups/assign`	`customerGroupIds`, `storeId`	`MageB2B_CustomCatalog::save`
`POST`	`/V1/customcatalog/catalog/:catalogId/customer-groups/unassign`	`customerGroupIds`, `storeId`	`MageB2B_CustomCatalog::delete`

Customer- and group-centric routes manage catalogs in the other direction:

Method	Endpoint	Body parameters	ACL resource
`POST`	`/V1/customcatalog/customer/:customerId/catalogs/assign`	`catalogIds`, `storeId`	`MageB2B_CustomCatalog::save`
`POST`	`/V1/customcatalog/customer/:customerId/catalogs/unassign`	`catalogIds`, `storeId`	`MageB2B_CustomCatalog::delete`
`GET`	`/V1/customcatalog/customer/:customerId/catalogs/:storeId`	—	`MageB2B_CustomCatalog::manage`
`POST`	`/V1/customcatalog/customer-group/:customerGroupId/catalogs/assign`	`catalogIds`, `storeId`	`MageB2B_CustomCatalog::save`

Method	Endpoint	Body parameters	ACL resource
`POST`	`/V1/customcatalog/customer-group/:customerGroupId/catalogs/unassign`	`catalogIds`, `storeId`	`MageB2B_CustomCatalog::delete`
`GET`	`/V1/customcatalog/customer-group/:customerGroupId/catalogs/:storeId`	—	`MageB2B_CustomCatalog::manage`

Customer and customer-group assignments are store-view scoped, so `storeId` is a required parameter on every customer and group route — in the request body for writes and in the URL for reads. Product membership belongs to the catalog itself and has no store scope.

Every `unassign` route returns the number of deleted assignment rows, and the `GET .../catalogs/:storeId` reads return an `int[]` of assigned catalog IDs. Assign routes return no payload. An HTTP success does not prove that every submitted ID was applied.

Create a Magento integration with only the Custom Catalog resources it needs. After a synchronization, verify one affected customer in the storefront and process partial failures rather than treating an HTTP success as proof that every assignment changed.

ACL resources

The module defines `MageB2B_CustomCatalog::all` with `::manage`, `::save` and `::delete` children, plus `MageB2B_CustomCatalog::config` for the configuration section. The same resource IDs protect the Admin: reads use `::manage`, mutating actions (save, assign, unassign, copy) are bound to `::save`, and deletes use `::delete`. Grant integrations and admin roles the smallest set that covers their operations.