Contents

WebRamos Google Merchant Feed – User Guide

Version: 1.2.0 **Compatibility:** Adobe Commerce / Magento Open Source 2.4.7 – 2.4.9, PHP 8.1 – 8.5 **Vendor:** WebRamos

---

1. Overview

WebRamos Google Merchant Feed publishes your Adobe Commerce and Magento Open Source catalogue to Google Merchant Center, so your products can appear in Google Shopping without a weekly spreadsheet export.

It gives you two ways to get the data there, and you pick one **per store view**:

	Scheduled Fetch	Merchant API push
How data moves	Google downloads a static feed file from a URL on your storefront	Magento sends each product to Google over Merchant API v1
Google credentials	None	Service account, Merchant Account ID, Data Source ID
Google Cloud project registration	Not needed	**Required, once, outside Magento** (see §2.3)
Freshness	Whatever schedule Google is set to fetch on	Minutes, driven by a queue and cron
Affected by Google API deadlines	No	Yes, and this extension is on Merchant API v1
Set up in	Minutes	An hour, mostly on Google's side

If you are starting out, or if you do not want to manage a Google Cloud service account, start with Scheduled Fetch. Nothing is lost by changing your mind later: the setting is a dropdown, and the attribute mapping is shared by both modes.

About the Google API surface

This matters more than it normally would, because two Google deadlines have already passed:

- **Merchant API v1beta** was shut down on **2026-02-28**.
- **Content API v2.1 for Shopping** was retired on **2026-08-18** and now returns `410 Gone`.

Extensions still built on either of those stopped working on those dates. This extension addresses **Merchant API v1** and never implemented Content API v2.1. Scheduled Fetch is unaffected by any API deadline at all, because Google simply downloads a file over HTTP.

Key features

- Two delivery modes, chosen per store view.
- Attribute mapping grid over 67 Google Merchant attributes, each pointing at a Magento product attribute or a fixed value.
- Admin dashboard, products grid with per-product feed exclusion and mass actions, Feed Preview, and an API log grid.
- Feed export as XML (RSS 2.0), JSON, or tab-separated CSV.
- Sandbox mode, which runs the whole pipeline without calling Google.
- Two command-line tools, including a dry run that prints the exact JSON body Google would receive.
- Cron: a full sync on a frequency you choose, and an incremental sync every ten minutes.

Compatibility

Platform	Versions
Adobe Commerce	2.4.7 – 2.4.9

Platform	Versions
Magento Open Source	2.4.7 – 2.4.9
PHP	8.1, 8.2, 8.3, 8.4, 8.5
Google API	Merchant API v1
Database	Two tables added (sync queue, API log)
Storefront	One route, which serves the feed file Google downloads. It renders no theme markup, so it is theme-independent

---

2. Installation

2.1 Via Composer

```
composer require webmaster-ramos/module-google-merchant
bin/magento module:enable WebRamos_GoogleMerchant
bin/magento setup:upgrade
bin/magento setup:di:compile     # production mode only
bin/magento cache:flush
```

`setup:upgrade` creates the two database tables the extension uses (a sync queue and an API log) and adds one product attribute, **Exclude from Google Merchant Feed**, which appears on the product form and in the product grid.

Three Google SDK packages are installed as dependencies: `google/auth`, `google/shopping-merchant-products` and `google/shopping-merchant-datasources`. They are installed even if you only ever use Scheduled Fetch.

2.2 Verify

```
bin/magento module:status WebRamos_GoogleMerchant
```

After enabling the extension you will find a new menu at **Marketing → Google Merchant Feed**, with Dashboard, Products, Feed Preview, Logs and Settings.

2.3 Before you can use API mode: register your Google Cloud project

Read this before configuring API mode. Merchant API v1 refuses calls from a Google Cloud project that has not been registered as a developer. This is a one-time action per Cloud project, performed by whoever owns that project, and it happens **outside Magento**:

`POST https://merchantapi.googleapis.com/accounts/v1/accounts/{ACCOUNT}/developerRegistration:regis`

Google documents it under “Register as a developer” in the Merchant API quickstart. Until it is done, **every** API call fails, no matter how the extension is configured, and the failures look like permission errors rather than like a missing registration.

This extension deliberately does not make that call. It lives in a different Merchant API sub-API, which would mean shipping an extra Composer dependency for something a merchant does once in the lifetime of a Cloud project.

Scheduled Fetch needs none of this.

---

3. Configuration

Navigate to **Stores → Configuration → Webmaster Ramos → Google Merchant Feed**, or use **Marketing → Google Merchant Feed → Settings**.

Almost everything here is scoped per store view. Use the scope switcher at the top left to configure each store view that sells to a different country or in a different language.

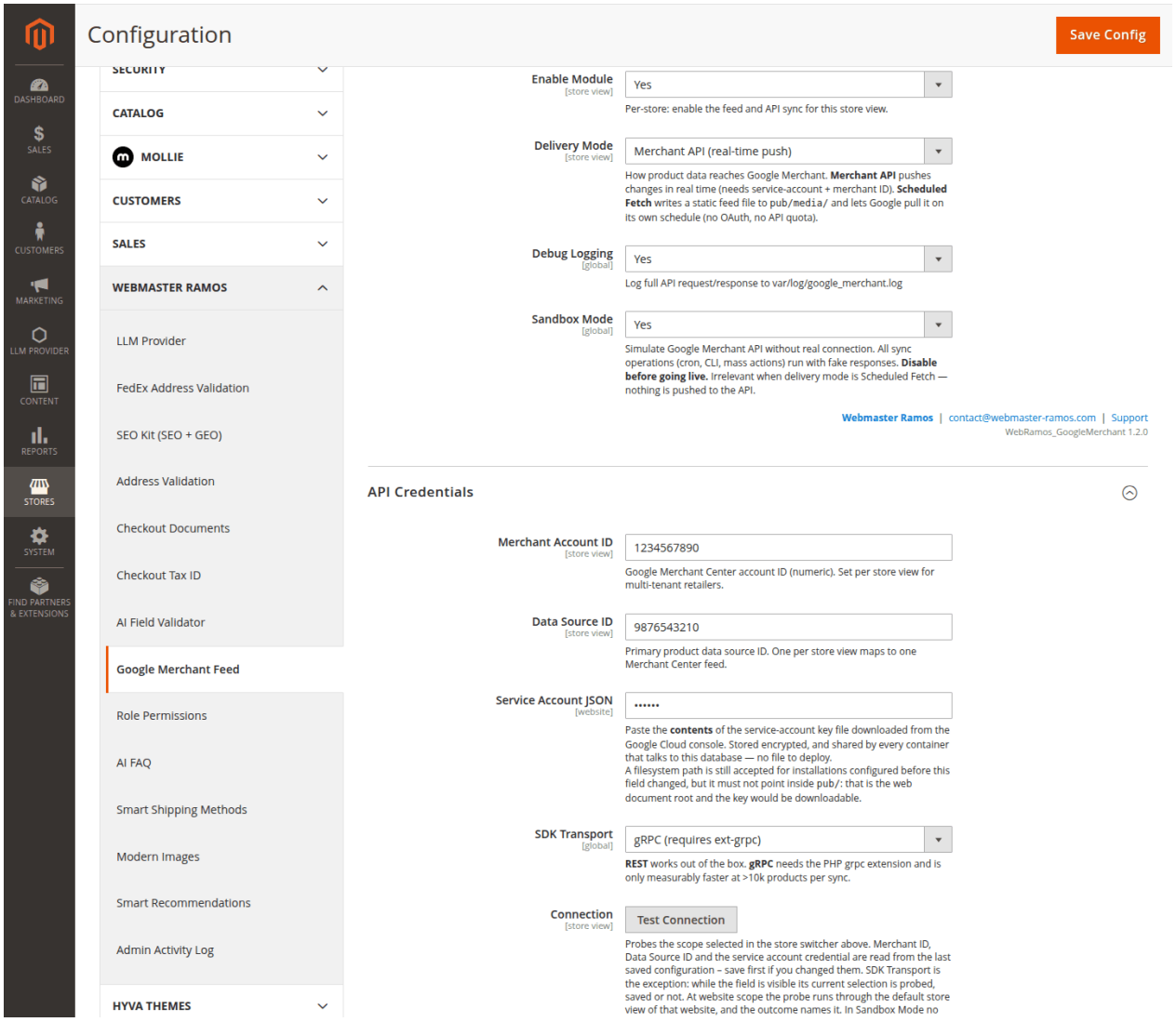

Figure 1: Configuration – General and API Credentials in Merchant API mode

3.1 General

Field	Description	Default
Enable Module	Enables the feed and API sync for this store view. Nothing runs until this is Yes.	No
Delivery Mode	**Merchant API** pushes changes to Google over the API. **Scheduled Fetch** writes a static feed file under `pub/media/` and lets Google pull it. This choice decides which of the groups below apply.	Merchant API
Debug Logging	Writes the full request and response body of every API call to `var/log/google_merchant.log`. Useful while setting up; turn it off afterwards.	No
Sandbox Mode	Simulates the Google API. Every sync operation (cron, command line, mass actions) runs with fabricated responses and nothing leaves your server. **Disable before going live.** Only shown in Merchant API mode, because Scheduled Fetch never calls the API.	No

The group ends with a support banner naming the extension and the installed version. Quote that line when you contact support.

3.2 API Credentials

Merchant API mode only. These four fields are hidden in Scheduled Fetch mode.

Field	Description
Merchant Account ID	Your Google Merchant Center account ID, numeric. Set it per store view if you run several Merchant Center accounts from one Magento installation.
Data Source ID	The primary product data source ID in that Merchant Center account. One data source per store view maps to one Merchant Center feed.

Field	Description
Service Account JSON	Paste the **contents** of the service-account key file downloaded from the Google Cloud console. It is stored encrypted and shown as asterisks. Since 1.1.0 this is content rather than a file path, which is what makes the extension work in containers: the credential travels in the database that web, cron and command line already share, so there is no volume to mount and no file permission to get wrong, and rotating the key is a paste rather than a rebuild. A filesystem path is still accepted for installations configured before this changed, but it must not point inside `pub/`, which is the web document root.
SDK Transport	**REST** works out of the box. **gRPC** needs the PHP `grpc` extension and is only measurably faster above roughly ten thousand products per sync.

The JSON is validated when you press Save Config: it must parse and carry a service-account type, a client email and a private key. If you paste OAuth client credentials by mistake, which is a common mix-up, you are told immediately rather than hours later in a cron log.

Setting it from a deployment script works the same way and stores ciphertext:

```
bin/magento config:set googlemerchant/api/service_account_json "$CREDENTIALS"
```

3.3 Feed Settings

Store Base URL, **Target Country**, **Content Language** and **Include Configurable Products** apply in both delivery modes. The rest apply to Scheduled Fetch only and are hidden in API mode.

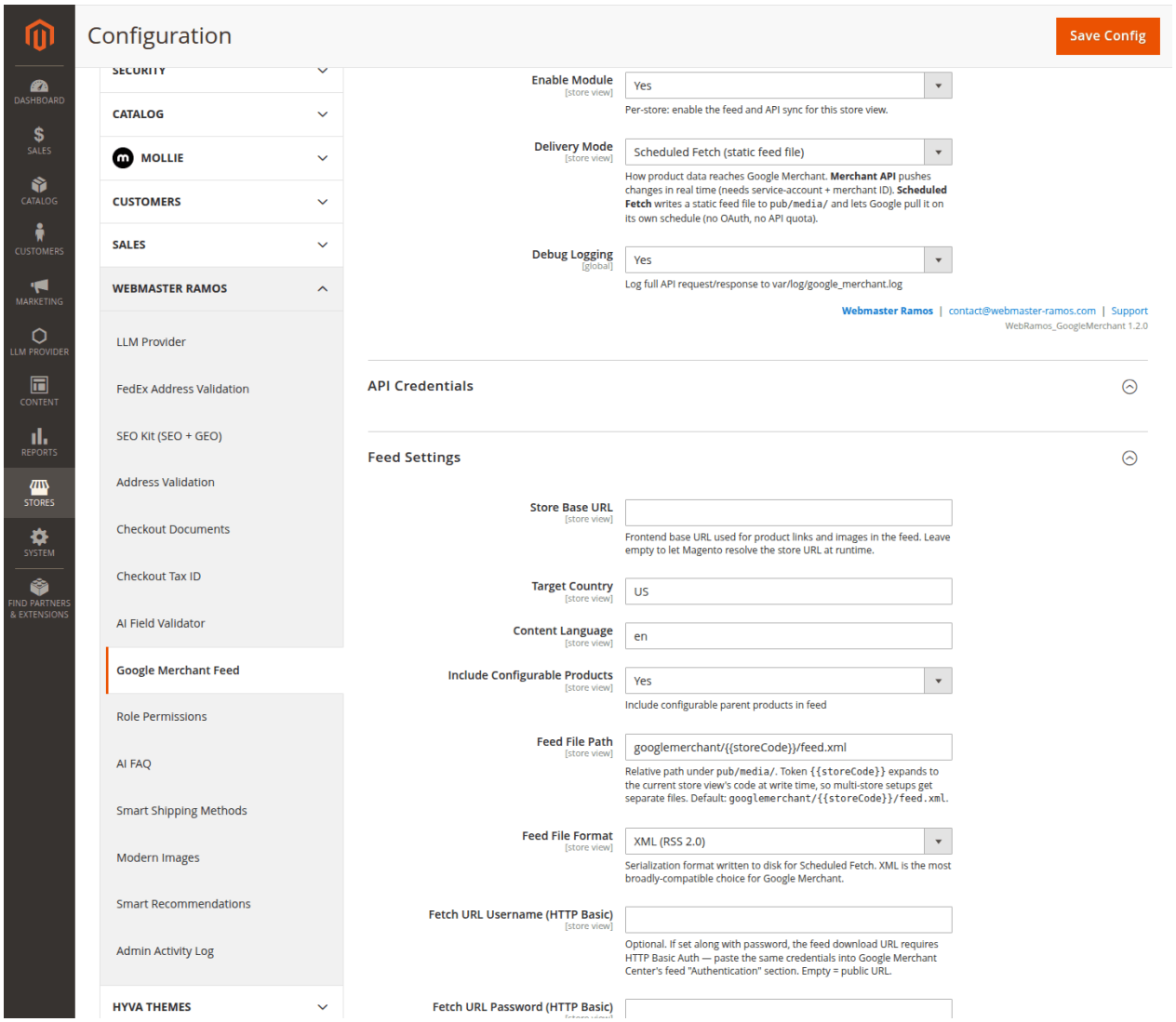

Figure 2: Configuration – Feed Settings in Scheduled Fetch mode, with API Credentials empty

Field	Description	Default
Store Base URL	The frontend base URL used for product links and image links in the feed. Leave it empty and Magento resolves the store's own URL at runtime. Set it only if the URL Google should see differs from the one Magento knows.	(empty)
Target Country	Two-letter country code for the feed.	US
Content Language	Two-letter language code for the feed.	en

Field	Description	Default
Include Configurable Products	Whether configurable parent products are sent alongside their simple children.	Yes
Feed File Path	Path relative to `pub/media/`. The token `{{storeCode}}` expands to the store view's code when the file is written, so a multi-store installation gets one file per store with no manual edits. Paths containing `..`, absolute paths and control characters are refused.	`googlemerchant/{{storeCode}}/feed.xml`
Feed File Format	`xml`, `json` or `csv`. XML is RSS 2.0 with Google's namespace and is the most broadly compatible choice for Merchant Center. CSV is tab-separated.	xml
Fetch URL Username (HTTP Basic)	Optional. Set together with the password to require HTTP Basic authentication on the download URL, and paste the same pair into the Authentication section of the feed in Google Merchant Center.	(empty)
Fetch URL Password (HTTP Basic)	Stored encrypted. Leaving both fields empty makes the feed URL public.	(empty)

3.4 Attribute Mapping

One grid, shared by both delivery modes. Each row maps a **Google attribute** to either a Magento product attribute or a **Fixed Value**. To use a fixed value, choose "Fixed Value" in the Magento Source column and type the value in the Fixed Value column. A row with only a Fixed Value filled in also works.

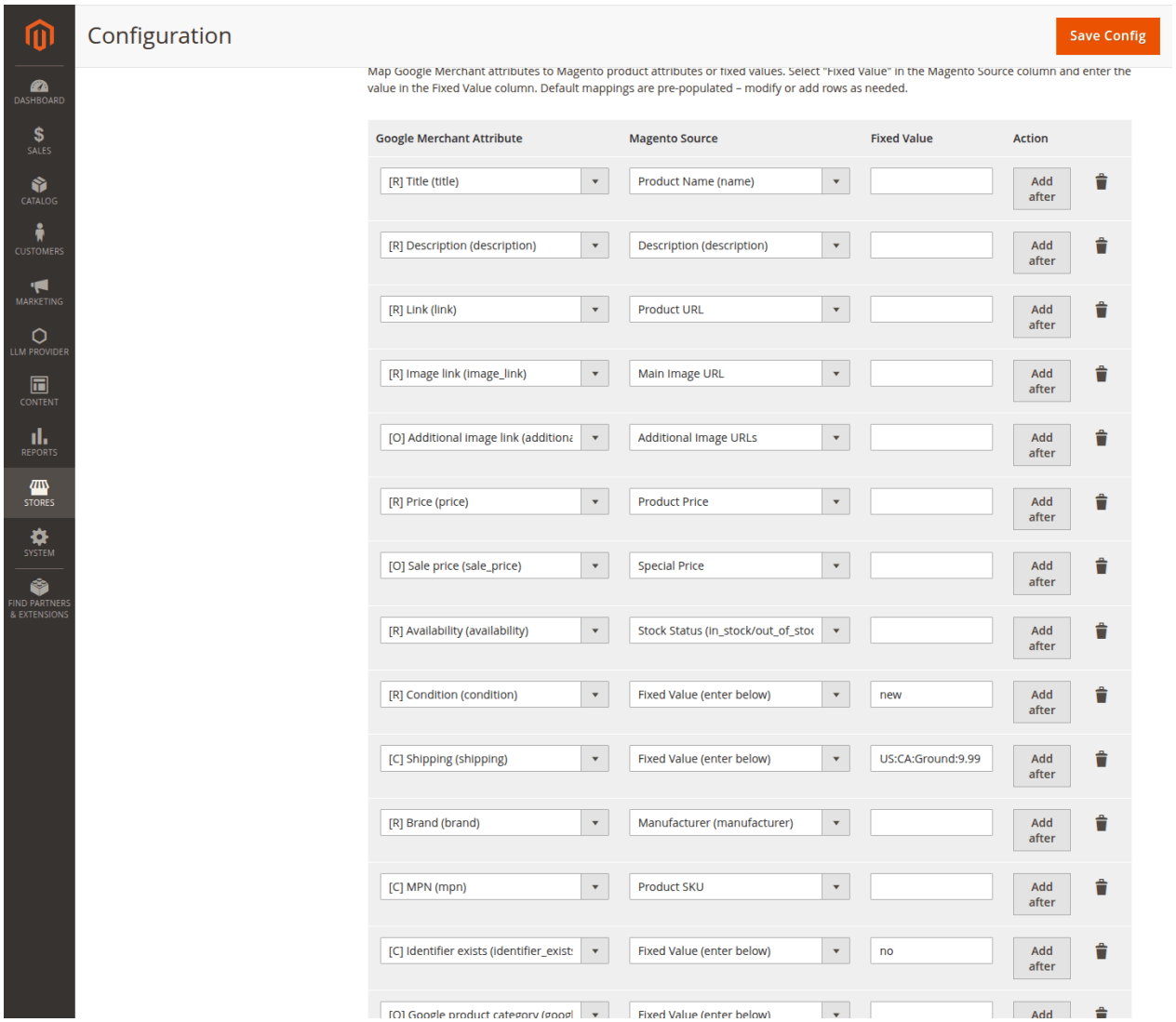

Figure 3: Configuration – Attribute Mapping, including a colon-delimited shipping value

Nineteen rows are pre-populated with sensible defaults, including special sources for the product URL, main image, additional images, price, special price, stock status, SKU and weight.

What the 67 attributes on offer actually do.

- **57 of them go straight through to the Merchant API** and into the feed file, each converted to the type Google declares for it: prices, dates, dimensions, measures, fixed-list values, whole numbers, yes/no values and comma-separated lists.
- **Six are multi-field messages** and take a colon-delimited value written in Google's own text-feed spelling:

Attribute	Shape	Example
`shipping`	`country:region:service:price`	`US:CA:Ground:9.99 USD`

Attribute	Shape	Example
`product_detail`	`section:attribute_name:value`	`General:Material:Leather`
`loyalty_program`	`program:tier[:price[:points[:interval]]]`	`myprogram:silver:10.00 USD`
`installment`	`months:amount[:downpayment[:credit_type]]`	`12:50.00 USD`
`subscription_cost`	`period:period_length:amount`	`month:12:29.99 USD`
`certification`	`authority:name:code[:value]`	`EC:EPREL:1234567`

`shipping`, `loyalty_program` and `certification` accept several groups separated by commas. `installment` and `subscription_cost` take one value only, so a comma anywhere in the cell refuses the whole cell rather than guessing which half you meant.

- **Three are feed-file only:** `tax`, `tax_category` and `ships_from_country` have no field in Merchant API v1 at all. They are written to the Scheduled Fetch file and are not sent over the API. This is stated rather than silently half-done.

A value that cannot be read is skipped, not guessed. If a cell does not parse as the type Google expects, the attribute is dropped for that product and a line goes into the API log naming the attribute, the value, and the shape that was expected. The rest of the product is still sent and the batch continues. Check the Logs grid after your first sync: this is where you find out that, for example, your Gender attribute's option labels read "Men" and "Women" while Google expects `male` and `female`.

3.5 Sync Settings

These apply to Merchant API mode. Scheduled Fetch regeneration shares the same frequency setting.

Field	Description	Default
Enable Automatic Sync (Cron)	Master switch for both cron jobs. With this off, nothing is pushed automatically; you can still export feeds and use Scheduled Fetch.	No
Full Sync Frequency (hours)	1, 3, 6, 12 or 24. Chosen here and translated into a cron schedule when you save, so you never edit cron XML.	6 hours
Batch Size	Products per sync batch.	50
Max Retry Attempts	How many times a failed product is retried before it is left alone.	3

Two cron jobs run:

- **Full sync**, on the frequency above. In Merchant API mode it enqueues every eligible product and processes the queue. For store views on Scheduled Fetch it rewrites the feed file instead.
- **Incremental sync**, every ten minutes on a fixed schedule. It picks up products saved since the last run and retries failures. It only does anything for store views in Merchant API mode.

Both are gated by **Enable Module** and **Enable Automatic Sync**. If either is off, nothing runs.

---

4. Setting up delivery

4.1 Scheduled Fetch, end to end

1. **General:** Enable Module = Yes, Delivery Mode = Scheduled Fetch.
2. **Feed Settings:** confirm Target Country and Content Language, and pick the Feed File Format. Leave the Feed File Path at its default unless you have a reason to change it.
3. Optionally set the HTTP Basic username and password so the feed URL is not public.
4. Save Config, then flush the cache.
5. Go to **Marketing → Google Merchant Feed → Dashboard** and press **Write Feed File Now**. The Scheduled Fetch table on that page then lists, for each store view: the public feed URL, the resolved path under `pub/media/`, and whether the URL is public or protected.
6. Open the public feed URL in a browser to confirm it downloads. It looks like this, with your storefront's own base URL:

   ```
   https://www.example.com/googlemerchant-feed/download/?store=default
   ```

7. In Google Merchant Center, create a Scheduled Fetch feed pointing at that URL, and paste the same username and password into its Authentication section if you set them.
8. From then on the cron job rewrites the file on the frequency set in Sync Settings, and Google fetches it on the schedule you gave Merchant Center.

4.2 Merchant API push, end to end

1. Confirm §2.3 is done: the Google Cloud project is registered as a Merchant API developer. Nothing below works until it is.
2. In Google Cloud, create a service account and download its JSON key. In Merchant Center, grant that service account access to the account.
3. **General:** Enable Module = Yes, Delivery Mode = Merchant API, and **Sandbox Mode = Yes** for now.
4. **API Credentials:** enter the Merchant Account ID and Data Source ID, and paste the service-account JSON. Save Config.
5. **Sync Settings:** Enable Automatic Sync = Yes, and choose a frequency.
6. Verify with a dry run and with sandbox syncs, as in §6.
7. Turn **Sandbox Mode** off, run one more sync, and check the Logs grid for real Google responses.

---

5. Admin screens

All five live under **Marketing → Google Merchant Feed**.

Screen	What it is for
Dashboard	Counters for eligible products, synced, pending, failed, skipped and queued; quick links; export buttons for XML, JSON and CSV; and the Scheduled Fetch table with each store's public URL, media path and authentication state.
Products	Every catalogue product with its image, **In Feed** state and **Sync Status**. Use the **Include in Feed** and **Exclude from Feed** mass actions to keep products out of the feed in bulk. The same override exists on the product edit form as **Exclude from Google Merchant Feed**, and it is scoped per store view.
Feed Preview	The resolved payload for each eligible product, with columns generated from your mapping rather than from a fixed list. Preview applies exactly the same eligibility rules as the real sync, so what you see here is what is sent.
Logs	Every API call, with action, SKU, status, message and timestamp. When Debug Logging is on, the full request and response bodies are stored too. This is where skipped attributes are reported.
Settings	A shortcut to the configuration section described in §3.

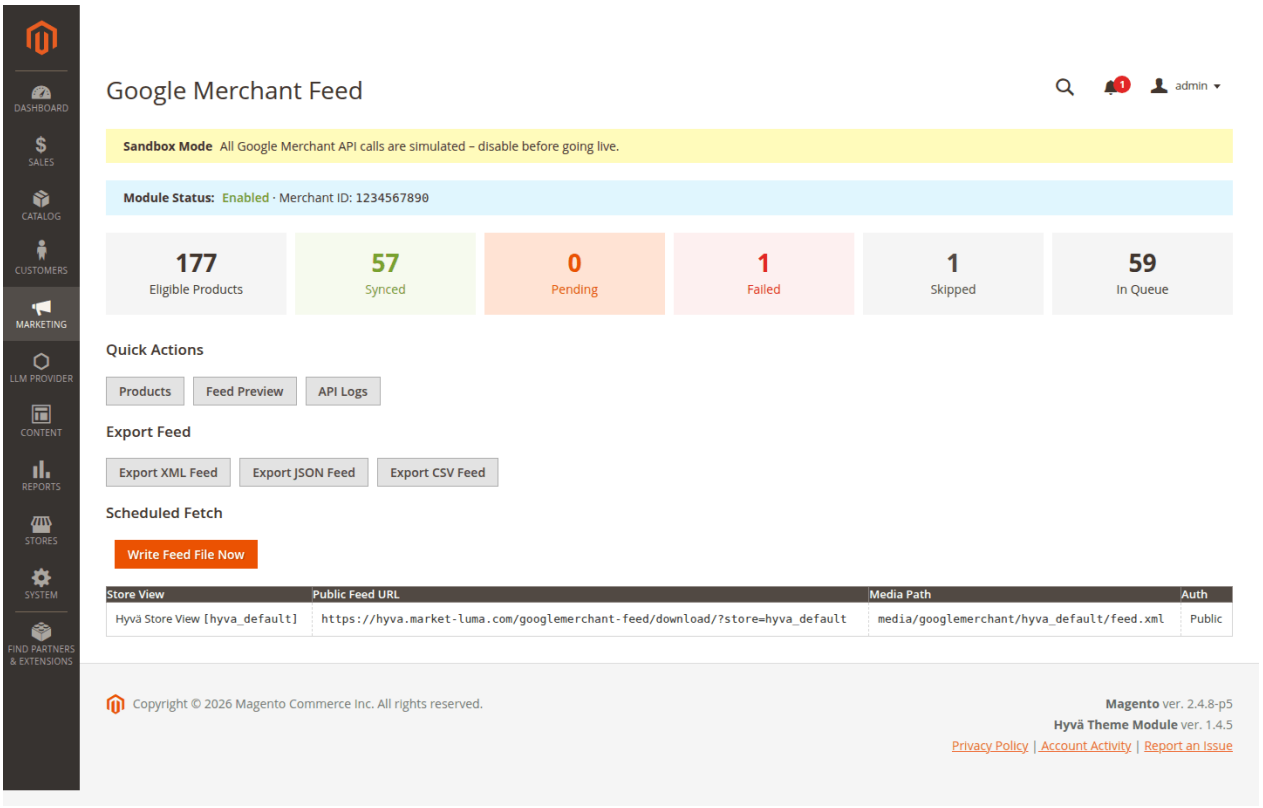

Figure 4: Dashboard – counters, quick actions, export buttons and the Scheduled Fetch table

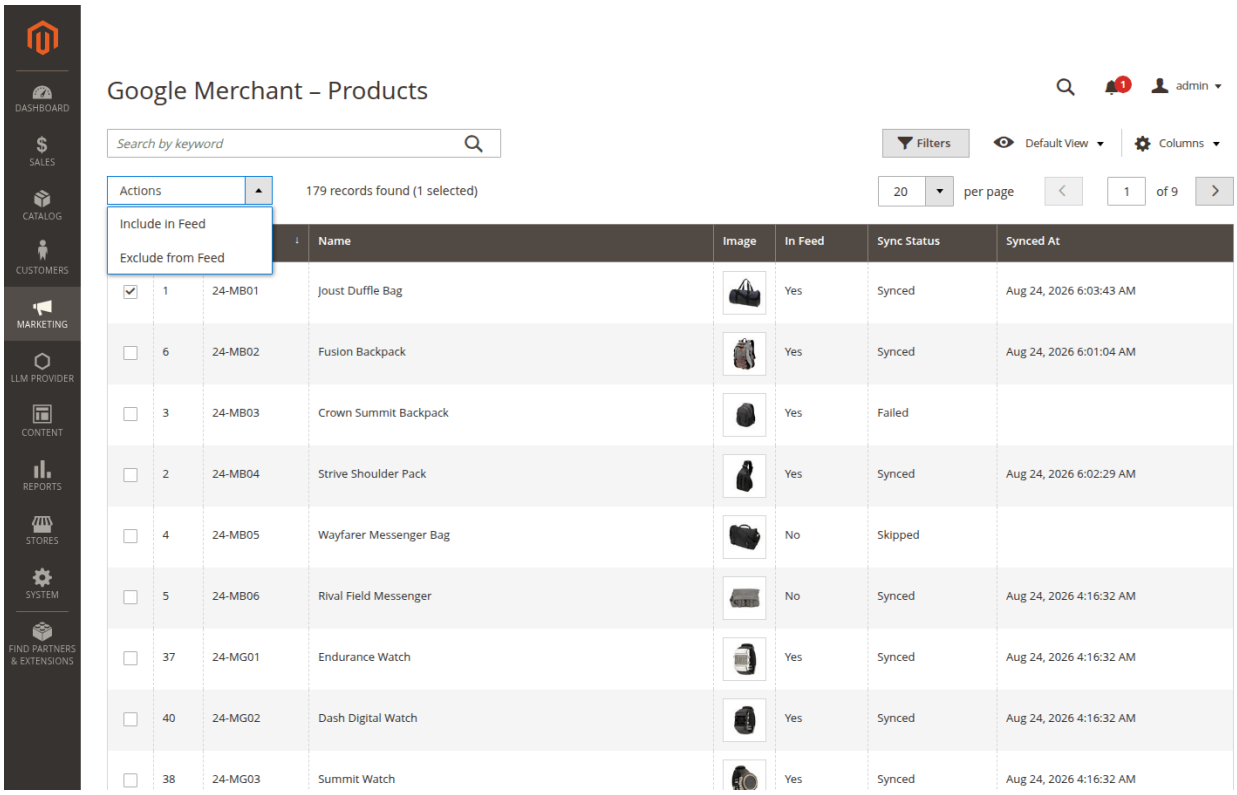

Figure 5: Products – In Feed and Sync Status columns with the mass-action menu open

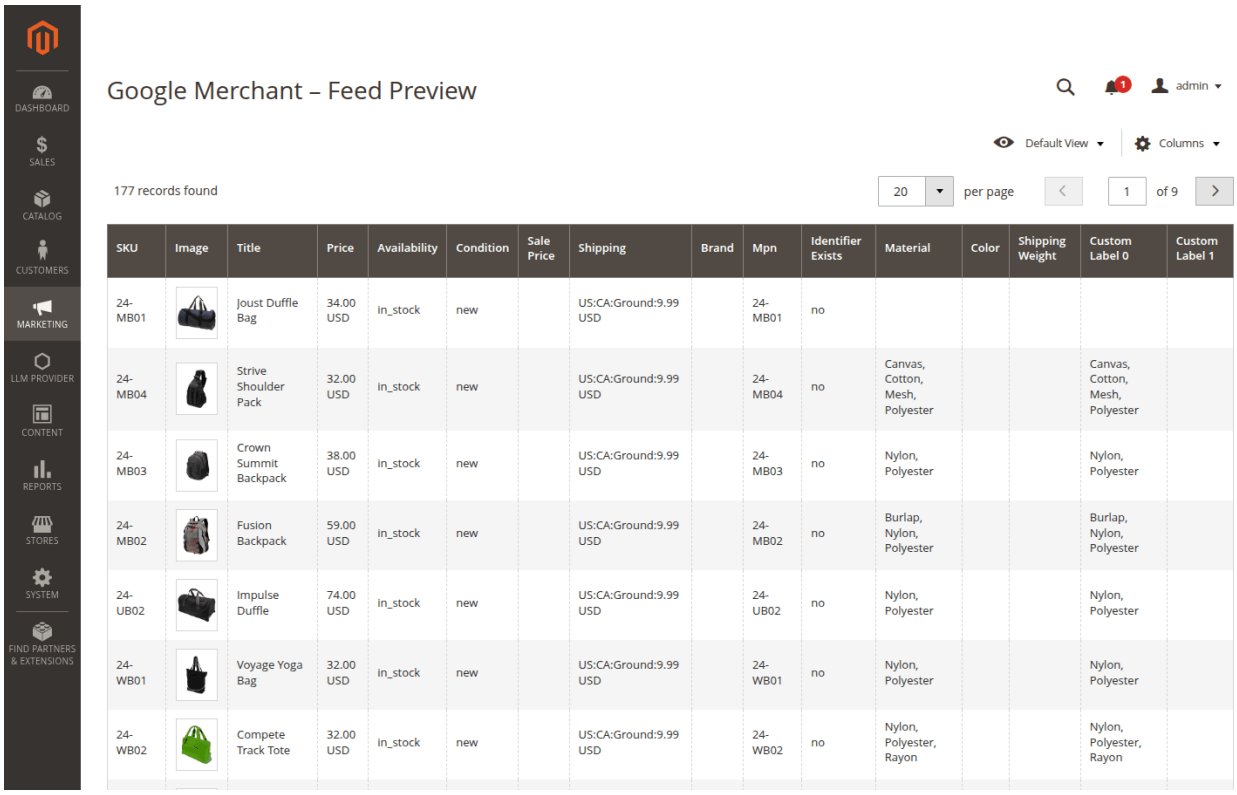

Figure 6: Feed Preview – columns generated from the mapping, not from a fixed list

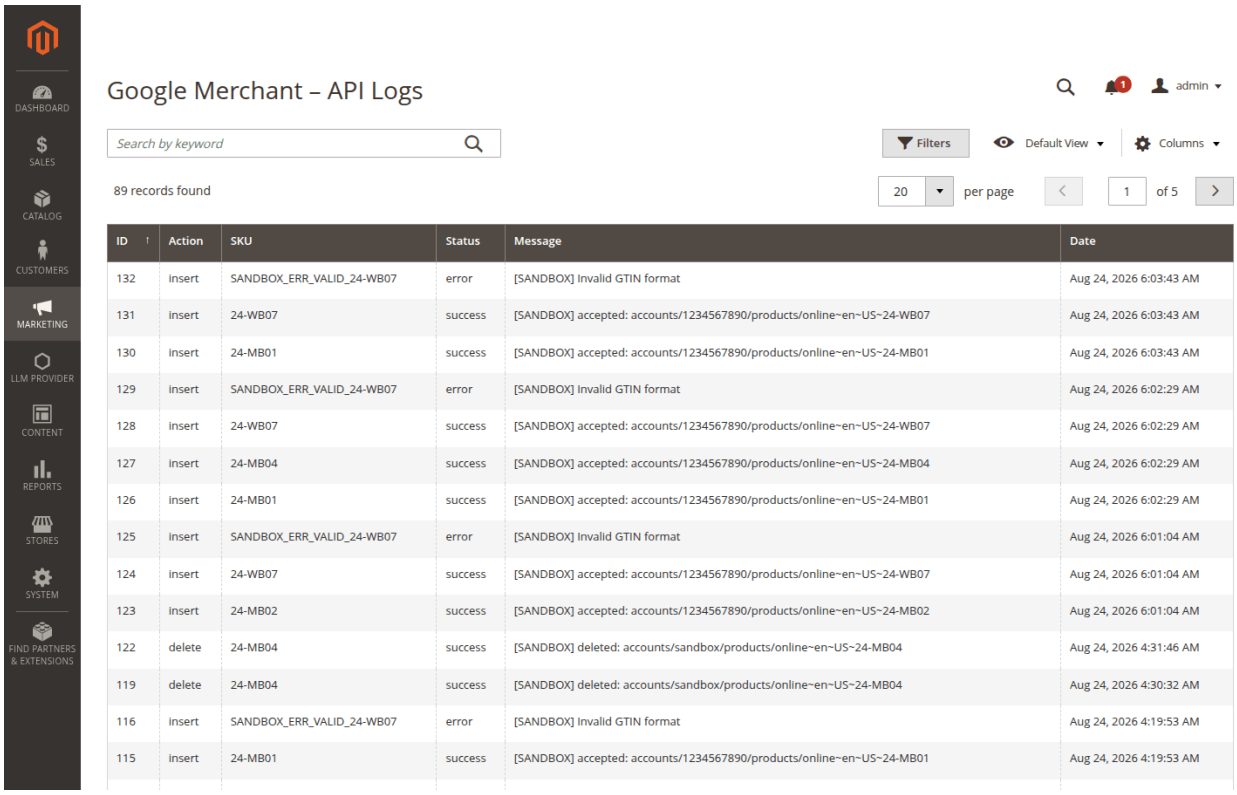

Figure 7: Logs – one row per API call, with a sandbox rejection above the successes

---

6. Command line

6.1 gm:sync – push products to Google

```
bin/magento gm:sync
```

Option	Description
`--sku=SKU`	Sync one product by SKU. Without it, the pending queue batch is processed.
`--dry-run`	Build and print the payload without calling Google.
`--limit=N`	Override the configured batch size for this run.
`--force`	Re-enqueue an already-synced row before processing. Requires `--sku`; without it the option is reported as ignored.

The command exits 0 on success and 1 on failure, so it is safe to use in a deployment script.

Figure 8: CLI – gm:sync –dry-run printing the exact request body

6.2 gm:export – write a feed file

```
bin/magento gm:export --format=xml
```

Option	Description
`--format=xml|json|csv` (`-f`)	Output format. CSV is tab-separated. Default: `json`.
`--sku=SKU`	Restrict to these SKUs. Repeatable. Default: every eligible product.
`--store=CODE`	Store view code. Default: the extension's default scope.
`--output=PATH` (`-o`)	Write to this file. Relative paths resolve from the Magento root. Default: stdout.
`--to-media`	Write to the store's configured Scheduled Fetch path under `pub/media/`. Requires `--store`, and overrides `--format` and `--output` because both come from that store's configuration.

`--to-media` uses exactly the same writer as the dashboard button and the cron job, so the admin, cron and the command line can never produce a different file.

---

7. Verifying that it works

Three checks, in the order that isolates a problem fastest.

7.1 The dry run: what would Google receive?

```
bin/magento gm:sync --sku=24-MB01 --dry-run
```

This prints the exact JSON body that product would be sent as: the bytes the SDK would put on the wire, not a re-rendering of them.

```
[DRY-RUN] 24-MB01 — request body:
{"offerId":"24-MB01","contentLanguage":"en","feedLabel":"US","productAttributes":{ … }}
[DRY-RUN] 24-MB01 (store 1) → pending
```

Two things make this the first check to run. It needs no credentials and opens no connection, so it works on a store that has never been configured. And it deliberately ignores Sandbox Mode: a dry run always shows what the real client would send, in either mode. When Google rejects a product, this output is what you compare against Google's complaint.

7.2 The feed URL: can Google reach the file?

For a store view on Scheduled Fetch, fetch the download URL the way Google will, from outside your network:

```
curl -I "https://www.example.com/googlemerchant-feed/download/?store=default"

# with HTTP Basic authentication configured
curl -I -u feeduser:secret "https://www.example.com/googlemerchant-feed/download/?store=default"
```

You want `200` and the right content type. A `401` without credentials is correct and expected when you have configured them. A `404` means the store view is not enabled, is not in Scheduled Fetch mode, or the file has not been written yet: press **Write Feed File Now** on the dashboard.

7.3 The Logs grid: what did Google say?

Marketing → Google Merchant Feed → Logs records every call. Read it after your first real sync, not only when something breaks. Successful syncs are recorded too, and so is every attribute that was skipped because its value could not be read, which is the single most useful thing in this extension for diagnosing a rejected product.

---

8. Troubleshooting

Symptom	Resolution
Every API call fails, with what look like permission errors	The Google Cloud project has almost certainly not been registered as a Merchant API developer. See §2.3. This is a one-time call to the `developerRegistration` endpoint made outside Magento, and until it is done every Merchant API v1 call fails regardless of how this extension is configured. It is by far the most common cause of "nothing works" on a fresh API-mode setup.

Symptom	Resolution
Google rejects products for missing required attributes	Google requires a title, description, link, image link, price, availability and condition on every item, and this extension checks the same set before sending. Open **Feed Preview** and look for empty cells in those columns. The usual causes are a product with no description, no image assigned, or a `condition` mapping row that was deleted (it ships mapped to the fixed value `new`).
Products are missing from the feed entirely	A product must be enabled, visible in the catalogue, priced above zero, and have a main image assigned, all four at once, and it must not be excluded. Check the **In Feed** column on the Products grid and the **Exclude from Google Merchant Feed** field on the product form, remembering that the override is per store view. Configurable parents appear only if **Include Configurable Products** is Yes.
Google disapproves items for missing GTIN or MPN	This is `identifier_exists`. Set to `no`, it tells Google the product has no manufacturer identifiers and suppresses GTIN and MPN validation. Set to `yes` (or left at Google's default, which is yes), validation is re-imposed and an item carrying neither identifier is disapproved. The extension ships this row mapped to the fixed value `no`. If your products genuinely have GTINs, map `gtin` to the attribute holding them and set `identifier_exists` to `yes`. `no`, `false` and `0` all mean no, in any capitalisation; an empty value leaves the attribute unsent, which is different from sending "no".
An attribute is set in the grid but never arrives at Google	Check the Logs grid. A value that cannot be read as the type Google declares is skipped with a line naming the attribute and the expected shape, rather than being guessed at. The usual causes are a price without a currency (`15.00` instead of `15.00 USD`), a date that is not ISO 8601, or an option label that does not match one of Google's fixed values.

Symptom	Resolution
Where does the feed file actually land?	Under `pub/media/` at the path in **Feed File Path**, with `{{storeCode}}` expanded. With the default template and a store view coded `default`, that is `pub/media/googlemerchant/default/feed.xml`. The dashboard's Scheduled Fetch table shows the resolved path for every store, which is quicker than working it out.
The feed URL returns 404	The store view is not enabled, is not set to Scheduled Fetch, or no file has been written yet. Check all three, then press **Write Feed File Now**.
The feed URL returns 401	HTTP Basic authentication is configured. Paste the same username and password into the Authentication section of the feed in Google Merchant Center, or clear both fields to make the URL public.
Nothing syncs automatically	Both **Enable Module** and **Enable Automatic Sync (Cron)** must be Yes, and Magento's cron must be running. Confirm with `bin/magento cron:run` and check the Logs grid afterwards.
The dashboard shows activity but Google shows nothing	Sandbox Mode is probably still on. It simulates every API call, so the queue advances and the log fills while nothing leaves your server. It is only visible in Merchant API mode.
Save Config rejects the service-account JSON	It has to parse as JSON and carry a service-account type, a client email and a private key. The most common cause is pasting OAuth client credentials, which are a different kind of key. Download the service-account key again from the Google Cloud console.
Save Config rejects the credential path	Paths inside `pub/` are refused, and rightly: `pub/` is the web document root, and a JSON key file placed there is downloadable over HTTPS with its full contents. Prefer pasting the JSON contents into the field, which is what the field expects since 1.1.0.

Logs: `var/log/google_merchant.log` when Debug Logging is on, plus the standard `var/log/system.log` and `var/log/exception.log`.

---

9. Support and changelog

- **Vendor:** WebRamos

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

When reporting an issue, include the identifier line from the end of the **General** group in **Stores → Configuration → Webmaster Ramos → Google Merchant Feed**, which names the extension and the version actually installed; your delivery mode; and the relevant rows from the Logs grid. For a rejected product, add the output of `bin/magento gm:sync --sku=<SKU> --dry-run`, which is the exact body Google was sent.