

AffStackDev Dynamic Pricing Complete Customer Reference

AffStackDev | Release 1.0.6 | Documentation revision 2026-10-21.1

Edition date: October 21, 2026.

Audience: Store owners, administrators and agency staff.

About this reference

AffStackDev Dynamic Pricing lets you create consumer pricing rules, select eligible products and customers, configure quantity offers and conditions, compare results, display current offers and explain retained order decisions. This reference covers every current setup and operation for store owners, administrators and agency staff, including the shopper-facing experience and recovery when a decision is held.

Use the contents and feature index to find the controls, requirements, results and limits for a task. The companion Master Technical Reference supplies exact calculation rules, fields, hooks, schemas and integration contracts.

What's new in 1.0.6

Release 1.0.6 follows 1.0.1 and adds an order discount, an amount or a percentage off the matching products together from a minimum spend, shared exactly over the order's lines ([An amount off the order](#)); rules that add to other offers, a box per rule that is off at the start ([Rules that add to other offers](#)); offers for shoppers who switch currency with WPML Multilingual & Multicurrency or FOX - Currency Switcher ([Shoppers in another currency](#)); and the offer price in the shop and on product pages with the old price crossed out, a shopper display that is off at the start ([Read offer prices in the shop and the cart](#)). The classic cart and the mini-cart now show a discounted line's old price crossed out, and the adjustment Fixed reduction per unit is called Fixed reduction: per unit, or once per cart for an order discount. Releases 1.0.2 to 1.0.5 were corrections listed in the plugin's changelog. Everything else in this section came with 1.0.1 and 1.0.0.5a.

Release 1.0.1 follows the launch release 1.0.0.5a. It changes nothing in how the plugin works: the guide built into the plugin now describes the current release, and version numbers have three parts from this release on. Everything else in this section came with 1.0.0.5a.

Release 1.0.0.5a was the AffStackDev launch release of Dynamic Pricing; WordPress showed it as version 1.0.0.5.1. It contains every change since 1.0.0.4.

Store credit and gift cards are payment. Applying store credit or a gift card never withholds or changes a pricing offer: the offer price comes first, then allowed ordinary coupons, then the credit. The **Allow recognized Store Credit with pricing offers** setting is kept but no longer affects pricing (see [Supported Store Credit](#)).

Purchases paid with credit count in history conditions. The part of a past order paid with store credit counts toward **Net paid order amount** and the order count, and a product paid with credit counts as previously purchased, so an order paid entirely with credit is no longer skipped; a refunded order counts no credit (see [Understand purchase-history conditions](#)).

Your license and updates. Each Dynamic Pricing license covers one live (Production) site and one test copy of it; an installation set to Staging, Development or Local uses the license's staging place. AffStackDev Suite needs no license key of its own. Download the plugin from **My Account → Downloads** on affstackdev.com; after activation, WordPress offers new releases through protected updates.

Contents

- [What's new in 1.0.6](#)
- [Connected AI actions](#)
- [Feature index](#)

- [Product purpose and setup](#)
- [Open the Dynamic Pricing workspace](#)
- [Access and retained records](#)
- [Use standard forms or enhanced navigation](#)
- [Find and manage saved rules](#)
- [Create or edit a pricing rule](#)
- [Add ALL/ANY conditions](#)
- [Understand purchase-history conditions](#)
- [Configure the engine and optional displays](#)
- [Interpret Integration status](#)
- [Run and interpret a sample cart](#)
- [Restore a previous revision or an archive](#)
- [Export configuration and import reviewed drafts](#)
- [Saved records and concurrent changes](#)
- [Read pricing information on a Woo order line](#)
- [Understand pricing explanations](#)
- [Recover from validation or request errors](#)
- [What the pricing engine changes](#)
- [Entering amounts and understanding rounded prices](#)
- [Common fields when creating a rule](#)
- [Bulk, graduated and set-price methods](#)
- [Adding conditions and handling missing history](#)
- [How units, limits and average prices are allocated](#)
- [Choosing between overlapping offers](#)
- [Running and interpreting a sample cart](#)
- [Reading retained explanations](#)
- [Why an offer was withheld and what to check](#)
- [How pricing reaches checkout](#)
- [Consumer eligibility](#)
- [Commerce Core and Commerce Bridge](#)
- [B2B connection and cleanup](#)
- [Retiring B2B before returning to consumer-only pricing](#)
- [Combining offers with coupons and Store Credit](#)
- [Shoppers in another currency](#)
- [Placing an order and reading its evidence](#)
- [Continuity status and retained rules](#)
- [Checkout integration boundaries](#)

- [Checkout messages and recovery](#)
- [Choose shopper displays](#)
- [Inspect an order's pricing decisions](#)
- [Use the pricing evidence report](#)
- [Use a merchant assistant proposal](#)
- [Resolve display and evidence problems](#)
- [Activate and manage the Dynamic Pricing license](#)
- [Read the license screen](#)
- [First activation or replacement](#)
- [Recover interrupted checks and understand status](#)
- [Reconnect a moved site or unreadable credential](#)
- [Release this site's activation](#)
- [Signed plugin updates](#)
- [What license data is retained](#)
- [Dynamic Pricing's Core connection status](#)
- [What Commerce Core does](#)
- [Read shared records and technical details](#)
- [Download reports and individual order records](#)
- [Customer eligibility and pricing integrations](#)
- [Shared cart lines and protected merchandise](#)
- [Accepted order history and publication recovery](#)
- [Diagnostic reports and integration interfaces](#)
- [Retained commerce records and safe updates](#)
- [Choosing a supported feature provider](#)
- [Shared configuration, pauses and ownership](#)
- [Fresh evaluation and short-lived permission](#)
- [One business operation, one retained identity](#)
- [Reconciling and recovering unfinished work](#)
- [Retiring a business policy safely](#)
- [Discovering checkout credit participation](#)
- [Shared operation troubleshooting](#)
- [Use the complete installed references](#)
- [Glossary](#)

Feature index

Capability or reference	Section
Accepted order history and publication recovery	Open reference
Access and retained records	Open reference
Activate and manage the Dynamic Pricing license	Open reference
Add ALL/ANY conditions	Open reference
Adding conditions and handling missing history	Open reference
An amount off the order	Open reference
Automatic retries and safe browsing	Open reference
B2B connection and cleanup	Open reference
Browse current offers	Open reference
Bulk, graduated and set-price methods	Open reference
Changed combination terms	Open reference
Checkout integration boundaries	Open reference
Checkout messages and recovery	Open reference
Choose shopper displays	Open reference
Choosing a supported feature provider	Open reference
Choosing between overlapping offers	Open reference
Combining offers with coupons and Store Credit	Open reference
Commerce Core and Commerce Bridge	Open reference
Commercial facts without price changes	Open reference
Common fields when creating a rule	Open reference
Configuration that remains with Dynamic Pricing	Open reference
Configure the engine and optional displays	Open reference
Consumer eligibility	Open reference
Continuity status and retained rules	Open reference
Create or edit a pricing rule	Open reference
Customer eligibility and pricing integrations	Open reference
Diagnostic reports and integration interfaces	Open reference
Discovering checkout credit participation	Open reference
Download a reference	Open reference
Download reports and individual order records	Open reference

Capability or reference	Section
Draft, publishing and committed	Open reference
Dynamic Pricing's Core connection status	Open reference
Entering amounts and understanding rounded prices	Open reference
Examples you can reproduce in Simulator	Open reference
Export configuration and import reviewed drafts	Open reference
Find and manage saved rules	Open reference
First activation or replacement	Open reference
Fresh decisions and integration changes	Open reference
Fresh evaluation and short-lived permission	Open reference
Fresh prices and the fallback page	Open reference
How older rule tiers work	Open reference
How pricing reaches checkout	Open reference
How units, limits and average prices are allocated	Open reference
Identity and ownership checks	Open reference
Inspect an order's pricing decisions	Open reference
Interpret each amount	Open reference
Interpret Integration status	Open reference
Interpreting integration diagnostics	Open reference
Legacy B2B and Dynamic Pricing	Open reference
One business operation, one retained identity	Open reference
Open and review the proposal	Open reference
Open the Dynamic Pricing workspace	Open reference
Placing an order and reading its evidence	Open reference
Prepare and install the plugin	Open reference
Preserve configuration and distinguish exports	Open reference
Product purpose and setup	Open reference
Read current cart savings and targets	Open reference
Read current integration status	Open reference
Read observations and current amounts	Open reference
Read offer prices in the shop and the cart	Open reference
Read pricing information on a Woo order line	Open reference

Capability or reference	Section
Read quantity offer examples	Open reference
Read rule and conflict decisions	Open reference
Read shared records and technical details	Open reference
Read the license screen	Open reference
Reading connection capabilities	Open reference
Reading retained explanations	Open reference
Recalculating and protecting the current total	Open reference
Reconciling and recovering unfinished work	Open reference
Reconnect a moved site or unreadable credential	Open reference
Recover an update that stops	Open reference
Recover from validation or request errors	Open reference
Recover interrupted checks and understand status	Open reference
Release this site's activation	Open reference
Resolve display and evidence problems	Open reference
Respond to connection messages	Open reference
Restore a previous revision or an archive	Open reference
Retained commerce records and safe updates	Open reference
Retiring a business policy safely	Open reference
Retiring B2B before returning to consumer-only pricing	Open reference
Run and interpret a sample cart	Open reference
Rules that add to other offers	Open reference
Running and interpreting a sample cart	Open reference
Save a paused draft deliberately	Open reference
Saved records and concurrent changes	Open reference
Selecting ordinary WooCommerce coupons	Open reference
Shared cart lines and protected merchandise	Open reference
Shared configuration, pauses and ownership	Open reference
Shared operation troubleshooting	Open reference
Shopper display freshness	Open reference
Shoppers in another currency	Open reference
Signed plugin updates	Open reference

Capability or reference	Section
Stores using Commerce Bridge	Open reference
Supported Store Credit	Open reference
Understand pricing explanations	Open reference
Understand purchase-history conditions	Open reference
Understand what each part controls	Open reference
Update deactivate and remove	Open reference
Use a merchant assistant proposal	Open reference
Use standard forms or enhanced navigation	Open reference
Use the complete installed references	Open reference
Use the pricing evidence report	Open reference
Use the references with AffStackDev AI	Open reference
What can enter a calculation	Open reference
What Commerce Core does	Open reference
What license data is retained	Open reference
What remains available during a license hold	Open reference
What the pricing engine changes	Open reference
Which cart items can receive an offer	Open reference
Why an offer was withheld and what to check	Open reference

Product purpose and setup

Dynamic Pricing creates consumer offers on eligible purchased products in a WooCommerce cart. You can build percentage reductions, fixed reductions, fixed unit prices, quantity breaks, graduated rates and fixed-price groups; select their products, customers, schedule and conditions; and inspect the result before enabling them. The plugin also explains current cart decisions, keeps order evidence, provides reports and can display fresh eligible offer examples to shoppers.

This reference covers release 1.0.1. Each chapter includes the controls, prerequisites, results, operating limits and recovery for its area. The technical reference specifies exact stored fields and integration payloads. Product version numbers and internal schema numbers identify different things; changing the plugin release does not mean existing rules are rewritten.

Prepare and install the plugin

Use WordPress 6.7 or later, WooCommerce 10.0 or later and a 64-bit PHP installation running at least PHP 7.4. The optional shared Commerce Core requires PHP 8.1 or later. A store relying on a Core-provided audience must satisfy that additional requirement.

1. Upload and activate the signed Dynamic Pricing installer in WordPress Plugins.
2. Open Settings → Dynamic Pricing license to connect this product's license.
3. Open AffStackDev Suite → Dynamic Pricing when Suite displays the admitted product, or WooCommerce → Dynamic Pricing when using standalone navigation.
4. Open Settings & integrations and choose the audience policy appropriate to the store. A store with B2B accounts needs the verified supported audience connection; selecting roles is not a replacement for that connection.
5. Create a draft rule. Set its adjustment, target products, qualification, schedule and limits using the rule chapters.
6. Use Simulator with Include draft rules to inspect the intended setup. Sample simulation and a current-customer simulation have different sources of customer facts, explained in the Simulator chapter.
7. Enable the intended rule and the pricing engine. Fresh checkout eligibility and the rule's current conditions still control each application.

A new installation starts paused. Updating an existing installation retains its settings and rules. If the required WooCommerce runtime or 64-bit PHP is missing, pricing services do not start and an administrator who can activate plugins sees the requirements notice. Correct the runtime requirement and reload the administration screen; changing a rule cannot resolve a missing runtime.

Understand what each part controls

Area	What it controls
Dynamic Pricing	Purchased-item unit-price offers, rule configuration, comparison, simulations and pricing evidence.
WooCommerce	Charged order lines, taxes, shipping, payments, coupons and actual refunds.
B2B audience provider	Authoritative business-account classification and consumer eligibility.
Free Gifts & BOGO	Gift and reward-item offers, including retained reward definitions that Dynamic Pricing does not execute.
Store Credit	Its own credit instruments, customer eligibility, redemption and financial ledger. Dynamic Pricing can permit specifically recognized combinations.
Optional Suite	The product's administration location and effective enhanced interface policy.
Shared Commerce Core	Compatible audience and commerce coordination services where available and selected.
Product license	Permission for new pricing work and updates, with separate access to retained records and license recovery.

One plugin's presence does not automatically authorize another plugin's offer. The integration chapters explain which combinations you can enable, which facts must be available and what the displayed hold reasons mean.

Read current integration status

Settings & integrations reports the product's audience, owner protection and companion integration state. Dynamic Pricing also exposes read-only status functions for compatible developer integrations. Those functions

report the product version, present shared runtime, customer eligibility diagnostics, credit compatibility, retirement state and continuity state.

The current cart decision is a result from this PHP request. It is not a reusable permission token and may be absent when no cart calculation has occurred. A recognized companion still needs to return fresh supported facts for the actual customer and operation. Automatic successor takeover remains false in this release; use the documented retirement and recovery controls when changing an expected provider.

Update deactivate and remove

Update the existing plugin in place with the new signed installer. Its rules, settings, archived revisions and WooCommerce order evidence remain stored. Deactivation clears the product's scheduled license validation. Uninstall also clears that event but preserves business records and the installation identity; it does not automatically release the remote license slot.

Use the licensing screen for the explicit connection and deactivation actions described in the licensing chapter. Export saved pricing configuration when you need its readable local records. The order inspector and reports explain retained order information even when pricing is paused by licensing. Retained configuration is not automatically moved to another pricing or rewards plugin.

Removing an expected audience or commerce provider can cause pricing to withhold offers until its authority is restored or the supported retirement process completes. Do not interpret disappearance of a companion's menu as a transfer of its authority.

Open the Dynamic Pricing workspace

Open **WooCommerce → Dynamic Pricing**, or the **Dynamic Pricing** product workspace in AffStackDev when Suite hosts the product. Your account needs permission to manage WooCommerce. The tabs are **Overview**, **Rules**, **Simulator**, **Order inspector**, **Reports**, **Import / export**, and **Settings & integrations**. **Manage license** opens licensing administration; that separate screen requires permission to manage WordPress settings.

Overview shows whether the pricing engine is enabled, how many working rules are enabled, and which overlap strategy is selected. A rule counted as enabled can still be waiting for its start date, past its end date, or unsuitable for a shopper or cart. The saved-rule count excludes archives but can include retained reward rules. **Current account eligibility** describes the account viewing the screen, not every customer. Use **Create a rule**, **Open simulator**, and **Review integration status** to move into the relevant task.

Dynamic Pricing changes eligible purchased-item prices. B2B accounts, including accounts paying retail prices, do not qualify for its consumer offers. Free Gifts & BOGO owns gift and selected reward campaigns. Coupon combinations begin off. Saved pricing rules can be prepared and simulated while the global engine is paused.

Access and retained records

Normal rule editing, settings, import and simulation need both WooCommerce-management permission and current Dynamic Pricing product access. If access is held, the workspace displays **Retained pricing records** instead of the normal editing tabs. You can still open **Order inspector**, read **Reports**, and use **Export saved pricing configuration**. This preserves access to saved definitions and order evidence without enabling new pricing work. Resolve the license through **Manage license**; repeatedly saving a rule is not a license recovery action.

The **Documentation** card under **Settings & integrations** provides the complete customer and technical Markdown references. It is also available on the retained-record screen. See [reference downloads](#). Order reporting, account eligibility and live pricing have their own checks even when you can open the workspace.

Use standard forms or enhanced navigation

All ordinary tasks work through the same server forms. With **Use enhanced navigation and in-place saves when the Suite allows them** enabled, compatible pages update in place. Suite can turn enhancement off globally. When enhancement is off, use the normal buttons and page loads; paste JSON for import instead of relying on the optional file chooser.

Enhanced forms announce **Working...** and disable submit buttons during a request. You can still edit fields while a response is pending. If you do, the newer input stays on screen instead of being replaced by an older response. A message warns that the request may already have saved its earlier values. Opening another view asks before leaving unsaved changes and waits for a pending save-like request to finish. Browser Back/Forward and Suite navigation use the same protection. Product searches and condition edits remain unsaved until **Save rule**.

If a save response cannot be confirmed, keep your input and open **Open saved version** in another tab. Compare the current saved definition and revision before retrying. A connection error does not prove nothing was saved. After a normal page submission, the product keeps returned input/messages in your temporary form state for 15 minutes; do not treat a bookmarked form-state URL as permanent storage. In enhanced mode, a filtered rules screen can have a simplified URL: reloading that URL may reopen the default rule list, so reapply its filter when needed.

The interface supports keyboard focus, labeled controls, current-tab indicators and expandable details. Hold Ctrl or Command to select multiple entries in a multiselect. On narrow screens the fields stack; wide result tables scroll horizontally. JSON details wrap and scroll. Opening the saved rule in another tab lets you compare it without discarding the form in front of you.

Find and manage saved rules

Open **Rules**. Enter part of a rule name or UUID in **Search name or UUID**, choose **Status**, and select **Filter rules**. This search does not search product names. There are 25 records per page, ordered by lower priority first and then record ID; use the nearby numbered page buttons.

Filter	What it includes
Working rules	All nonarchived definitions, including retained reward definitions.
Enabled	Enabled ordinary pricing rules, even when their schedules are not currently active.
Draft / paused	Disabled ordinary pricing rules.
Archived	Archived definitions.
Retained rewards	Reward definitions that belong to Free Gifts & BOGO.
Everything	All retained definitions matching the search.

Each ordinary rule shows its UUID and revision. An enabled rule can display **Scheduled** before its start or **Expired** at its end; these badges do not change its saved enabled checkbox. Select the name to edit a working rule or view an archive's history. **History / restore** opens its previous revisions. **Duplicate as draft** opens an unsaved copy with a new identity, the same priority and settings, a name ending (copy), and Enable off. Review it and save it before it exists as a separate rule.

To manage several rules:

1. Check **Select rule** on each intended row.
2. If changing priority, enter the desired number beside each selected rule.
3. Choose **Action for checked rules: Pause, Enable, Save entered priorities, Archive and pause, or Restore as drafts.**
4. Select **Apply to checked rules** and read every row result.

Pause is the initial action. Priority is 0–9999; lower values win Priority mode and break ties. Only selected rows change. Each row saves independently against its displayed revision, so one conflict does not undo other successful changes. A submission accepts up to 50 selections, although a normal page shows 25. Enabling also rechecks current product/term/role references and currency. Restore leaves a rule paused; review and enable it separately. Archives cannot be edited or reprioritized until restored.

The product permits 100 working rules and 1,000 retained records including archives. Archive an unused ordinary rule to free a working slot. There is no delete command. Retained reward definitions also count toward storage limits. A **Paused — Free Gifts & BOGO feature** record has **View retained rule** and expandable details, but no mutation checkbox or duplicate action. Its definition remains exportable; Dynamic Pricing does not execute or automatically move it into Free Gifts & BOGO.

Create or edit a pricing rule

Choose **Create rule**, or expand **Start with an editable template**. The four templates are:

Template	Starting values to review
Simple percentage offer	10% off all eligible matching products.
Bulk quantity breaks	5% from quantity 3, 10% from quantity 5, and 15% from quantity 10.
Graduated quantity pricing	No base discount; 10% starts at unit 5 and 15% at unit 10.
Fixed price for a complete set	Any complete group of 3 eligible units for a total of 30 in store currency.

Templates open unsaved drafts. They exclude sale products and count across matching products. They do not save, enable or publish an offer automatically.

To prepare a rule:

1. Give it a name and choose the adjustment, value and currency.
2. Choose **Apply to**, select the intended products or terms, and set exclusions and any consumer roles.
3. Set quantity/spend minimums, price floor, priority and any quantity breaks.
4. Set the schedule and **Offer method**. For set pricing, choose Fixed unit price and enter the group's total in Value.

5. Add any extra conditions. Use product/attribute refresh buttons as needed; they do not save the rule.

6. Leave **Enable this rule** off while preparing, then select **Save rule**.

7. Run the saved draft in **Simulator** with draft inclusion. Return to the editor and enable it when its results are suitable. The global engine must also be enabled under Settings & integrations for live application.

A clean rule starts at 10% discount, all eligible purchased products, matching-cart quantity counting, minimum quantity 1, priority 100, sale exclusion on, no schedule, no extra restrictions and Enable off. The complete fields are:

Control	How to use it
Rule name	Required, at most 190 bytes after HTML tags are removed. It can be shown to shoppers when labels are on.
Adjustment	Percentage discount; Fixed reduction: per unit, or once per cart for an order discount; or Fixed unit price.
Value	Percent up to 100 with two decimal places; otherwise a money amount with up to six decimal places. Use digits and a decimal point, without symbols or thousands separators. A fixed set uses this as the total for the complete group.
Currency	Defaults to this store's currency. A differently named currency does not convert and will not match this store. Import and list activation require the store currency. Shoppers who switch currency with a supported switcher get the rule converted (Shoppers in another currency).
Enable this rule	Off initially. Enabling an individual rule is separate from enabling the engine.
Apply to	All eligible purchased products; Selected products / variations; Selected categories; Selected tags; Selected brands; Selected attribute values. Choose the corresponding selection below.
Find products by name or exact SKU; Find products	Refresh available products while retaining current selections and unsaved input. Up to 100 search matches appear, plus selected products and an exact SKU result. Narrow the search when necessary.
Products / variations	Select the targeted products or concrete variations for product scope.
Categories	Select categories for category scope; descendants are included in matching.
Exclude products / variations	Remove listed products from the rule before qualification.
Consumer roles (empty means any)	Limit an already eligible consumer offer to selected roles. This never makes a B2B customer eligible.
Exclude products currently on sale	On initially. Clear only if the offer should also consider products currently on sale.
Count quantities	Across eligible matching products, or Within each cart line.
Minimum quantity	Whole number 1–10,000; initially 1.
Minimum matching spend	Initially 0; eligible matching catalog amount before the offer.
Minimum unit price	Initially 0; the offer cannot reduce a unit below this amount or raise an existing price. Coupons applied later can reduce the payable amount further.
Priority	0–9999; initially 100. Lower numbers win in Priority mode and break ties.

Control	How to use it
Quantity breaks (optional)	Up to 20 unique lines, such as 5 = 10. Quantities are whole 1–10,000. Value means the chosen percentage, reduction or price. Blank lines are ignored; the saved order is numeric.
Starts / Ends	Site-local date and time. Start is inclusive and end exclusive. Leave blank for no limit; when both are set, end must follow start.
Offer method	Bulk: one qualifying rate for all matching units; Graduated: rates begin at their numbered unit; Mix and match: complete groups for a set price.
Units per qualifying group	Initially 3; whole 2–10,000, used for set offers.
Maximum groups per cart	Initially 0, meaning no additional cap. Whole 0–10,000; a nonzero value is only valid for grouped offers.
Maximum discounted units per cart	Initially 0, meaning no additional cap; whole 0–10,000. Set offers use complete groups only.
Maximum rule discount per cart	Initially 0, meaning no additional cap. Before coupon deductions; a set offer must fit as a whole.
Tags / Brands	Choose existing product tags or the native product-brand taxonomy when the matching Apply to option is selected.
Attribute / Attribute values	Select a registered global attribute, choose Refresh attribute choices without saving , then select its values. Variation matching uses its selected value, not every parent value.
Exclude categories / Exclude tags	Remove these terms' matching merchandise before qualification.

Each product/term selection accepts at most 100 submitted entries. Roles accept at most 30. Money amounts are nonnegative and at most 9,999,999.999999. Bulk percentage and fixed-reduction base values must be positive; graduated rates may start at zero, fixed prices may be zero, and a quantity tier may also use zero. Set offers require Fixed unit price and do not accept quantity breaks. Leave Maximum groups at zero for bulk/graduated rules even though that field remains visible.

Schedules use the WordPress site timezone displayed beside the fields. Saved schedules are instants; changing the site's timezone changes their displayed local time. A deleted product or term can make a save fail; choose an existing replacement or remove the stale selection. A role/history reference may still be syntactically saved after its underlying record changes, so use current choices and account simulation to verify qualification.

Graduated and set offers allocate units in a defined product/variation/cart-line order and share an average price across each Woo order line. An incomplete set is not a complete group. A conflicting offer cannot retain just part of a set. Refunds follow the recorded Woo line amount, including subsequent coupons and tax, rather than assigning a different physical item to a past graduated rate. See [pricing calculations](#) for the complete calculation examples.

Add ALL/ANY conditions

Open **Additional conditions** in the editor. These conditions supplement the rule's existing minimums, targets, exclusions, schedule and audience. The root starts with **ALL conditions in this group**. Choose **ANY condition in this group** when any one child should satisfy that group.

Use **Add condition** for a leaf and **Add group** for a nested ALL/ANY group. **Remove condition** and **Remove group** remove only the selected part of the form. The root cannot be removed. Every action returns the form with a reminder to save; nothing changes live until **Save rule**.

Condition	Enter
Eligible cart quantity	Whole 0–10,000 across eligible catalog merchandise.
Eligible cart catalog amount	Nonnegative catalog merchandise amount before pricing.
Matching quantity	Whole 0–10,000 within this rule's matching merchandise.
Matching catalog amount	Nonnegative amount within this rule's matching merchandise.
Customer role	Exact role slug, such as customer; use Equals/includes or Does not equal/excludes.
Customer type	guest or registered; use Equals or Does not equal.
Retained paid order count	Whole 0–10,000 in the requested lookback.
Net paid order amount	Nonnegative amount in the requested lookback.
Previously purchased product / variation	Positive product or variation ID; use Equals/includes or Does not equal/excludes.

Numeric fields support At least, At most, Equals, Does not equal, More than, and Less than. A newly added condition starts at At least, so change it for role, customer type or prior-product checks. **History lookback days** is 0–3650: 0 means all retained history. It does not change non-history conditions. Blank condition rows are ignored when saved.

A group has at most 12 children. The form allows two nested group levels beneath the root and leaves through the next level, with an overall save limit of 32 visited condition/group rows, including blank rows before they are discarded. If the tree is full, remove or simplify conditions. An unavailable fact is not treated as zero. A known matching branch can satisfy ANY, and a known failing branch can fail ALL; otherwise unresolved required facts keep the offer unavailable.

Understand purchase-history conditions

Purchase history uses the signed-in customer's retained Woo orders. It does not match guest orders by email or invent a history for a sample registered visitor. Use **Simulator → Current customer eligibility and roles** and choose the actual account to examine history-based conditions.

The reader considers processing, completed and refunded orders with a paid date that is not in the future, in the store currency; an order a shopper paid in another currency through a supported switcher counts converted back at its recorded rate (Shoppers in another currency), and one without a recorded rate is left out. It subtracts refunded value; an order with no positive net amount does not count. Net paid amount is the remaining whole-order total, which can include tax, shipping and other components already in Woo's total. Previously purchased products count only where both unrefunded quantity and unrefunded item value remain positive; both parent and variation IDs can qualify. Store credit and gift cards are payment: the part of an order paid with them counts toward its net amount and its items' value, so an order or product paid entirely with credit still counts. A refunded order counts no credit, because Store Credit returns it on a full refund. These are retained paid records, not a separate gateway-settlement ledger.

At most eight distinct history lookback windows can be requested across the considered rules. More than 500 returned orders or 5,000 inspected items makes history unavailable rather than partially counting it. Other-currency orders are skipped after the bounded order query, so a large retained account can reach the read limit even when fewer orders remain useful. If history cannot be confirmed, fix inaccessible records or narrow the requested lookbacks and run the account sample again. Do not assume unavailable history means a new customer.

Configure the engine and optional displays

Open **Settings & integrations** and review every control before selecting **Save settings**. New installations start with the pricing engine off, verified audience required, offer labels and enhanced navigation on, and coupon/credit combinations and shopper displays off.

Control	Choices and use
Enable the pricing engine	Turns live pricing work on or off; saved definitions/evidence remain.
When rules overlap	Compare offer savings (default) or First rule by priority wins. Savings comparison follows priority order and replaces whole conflicting offers only for a greater total reduction; it is not a search of every possible combination. Use Simulator for overlaps.
Customer eligibility	Require a verified audience provider (default), or This store serves consumers only. Use the declaration only for a consumer-only store; it does not override B2B or reset an expected integration.
Show the applied offer name in cart item details	On initially. Shows the selected rule name; does not remove order history when off.
Use enhanced navigation and in-place saves when the Suite allows them	On initially, subject to Suite's preference.
Ordinary coupon codes allowed with pricing offers	One existing supported ordinary code per line, at most 50 and 8,192 input bytes. Empty means pricing yields to ordinary coupons. Individual-use, virtual and plugin-managed coupons cannot be allowed here.
Allow recognized Store Credit with pricing offers	Off initially. Store Credit and gift cards are payment: applying them never withholds or changes pricing offers, whether this box is on or off. The setting is kept for existing integrations and has no pricing effect. Store Credit keeps its own customer/value/restriction checks.
Show quantity offer examples on product pages	Off initially. Examples use only that product's sample cart, before coupons/shipping.
Product display position	Before the add-to-cart form (default), After the add-to-cart form, or Shortcode placement only.
Show verified pricing savings in the cart	Off initially. Displays a verified current merchandise-pricing reduction.
Show next quantity / spend target in the cart	Off initially. Shows a current bounded opportunity, not a guaranteed cheapest cart.
Enable the current offers page shortcode	Off initially. Enables the supported offer browser where the shortcode is placed.

Control	Choices and use
Show the offer price in the shop and on product pages, with the old price crossed out	Off initially. Where a rule lowers the price of one more piece in the shopper's current cart, the shop, product grids and the product page show the normal price crossed out and the offer price. Pages stay the same for every visitor; the offer price is filled in privately (offer prices).

For an offers page, add `[affstackdev_pricing_offers]`. For manual product placement use `[affstackdev_pricing_table]`; for a Cart block/manual summary location use `[affstackdev_pricing_summary]`. Enabled classic-cart displays get automatic placement. The shopper chapters explain [product examples](#), [cart displays](#) and [offers pages](#), including when no result is shown.

Unit pricing is calculated before allowed Woo coupons; Store Credit then keeps its own allocation. The pricing floor does not promise a minimum final amount after those deductions. Allowing a coupon does not override that coupon's limits. Changing an ordinary coupon's properties or removing it can make a formerly selected combination unavailable. See [coupon and credit behavior](#).

Save settings checks the revision you opened. If another person changes settings first, your input is retained and you must compare with the current saved version before trying again. Provider expectations are separate retained records: changing Customer eligibility, importing a rule, or removing a plugin does not erase them.

Interpret Integration status

The **Integration status** panel reports the saved configuration and current request. It does not offer override controls. If a save failed, its report can describe saved settings while the form above still shows your unsaved input.

Row	What to check
Suite navigation	Whether this product uses Suite or the standalone WooCommerce menu.
Enhanced interface	Whether enhancement is effectively enabled or standard PHP forms are in use.
Commerce Core	Connected, absent, or expected/incompatible. Restore an expected connection when it is unavailable.
Legacy Commerce Bridge	Connected line protection, absent and unnecessary for ordinary pricing, or expected protection unavailable.
B2B discount cleanup	Whether B2B accepted the consumer-pricing cleanup registration. An unconfirmed registration holds pricing.
Current account eligibility / Audience decision owner	This administrator's reason and confirmed owner; not a global customer guarantee.
Purchased-item pricing	Current pricing continuity state and any hold.
Gift control	Whether Free Gifts & BOGO is loaded; it continues to own its gifts.
Shared feature takeover	Informational statement that saved rules remain with their current products.
Paid licensing	Legacy fixed information. Use Manage license for the actual current license state.

Row	What to check
B2B compatibility	Accepted declared compatibility or the specific connection problem.
Store Credit recognition	Recognition and validation readiness; settings and the particular instrument still govern combination.
Core runtime host	Selected host's status, identity and version.
B2B policy retirement	Whether current accepted removal proof exists. Only explicit consumer-only pricing can use it.
Calculation order	Pricing first, allowed coupons next, Store Credit allocation later; Woo owns tax/shipping.

If a once-connected audience or line-protection owner disappears, restore it or follow its supported retirement procedure. Selecting consumer-only alone is not that procedure. See [audience recovery](#), [retirement](#), and [Core hosting](#).

Run and interpret a sample cart

Save the rule first, then open **Simulator**. It can examine saved drafts while the engine remains paused; it does not include unsaved editor changes.

1. Search by name or exact SKU with **Find sample products**. Select 1–20 simple products or individual variations; do not select a variable parent.
2. Set **Quantity of each selected product**, from 1 to 500. The same quantity applies to every selected product.
3. Choose **Audience source**. For a hypothetical case choose **Sample audience and roles**. For actual current eligibility/roles/history, choose **Current customer eligibility and roles**.
4. In account mode, use **Find a customer by name, login or email**, choose **Find customers**, and select **Customer for account lookup**. Guest is ID 0. Search returns up to 50 current-site users and retains an existing selected member.
5. In scenario mode choose **Sample customer type** (Guest or Logged in), **Sample audience** (Eligible consumer, B2B including 0%, or Unknown), and optional **Sample roles**. These do not verify a real person or provide purchase history.
6. Select **Include draft rules in this simulation** if appropriate. Archives and retained rewards remain excluded.
7. Optionally select **An additional unapproved coupon is applied**, or enter codes in **Coupon codes for a combination policy check**. The latter checks combination permission only.
8. Select **Run simulation** and review **Price explanation**, all reasons, every product line and the rule/conflict details.

The result shows **sample merchandise reduction**, base and offer prices per unit, quantity and the selected rule or reason. Account lookup shows eligibility and its owner; a scenario explicitly says that no customer eligibility was verified. A history warning means history conditions cannot be confirmed. A hypothetical Logged in selection does not change that.

The sample does not calculate final tax, shipping, payment totals, coupon deductions, Store Credit spending or gift-engine outcomes. It uses current catalog prices and the saved overlap strategy, rather than changing a live cart. Large samples are bounded: in addition to 20 products and 500 units each, product count × quantity ×

working rule count must fit 200,000. Reduce the sample if requested. The technical [simulator reference](#) explains its exact inputs and result fields.

Restore a previous revision or an archive

Open **Rules → History / restore** for the record. **Current saved definition** shows the current JSON. Expand a previous revision to inspect its saved definition, flags, priority, timestamps and the account that replaced it. Up to 50 previous revisions remain; this is rolling history rather than every change forever.

Choose **Restore this revision as a draft** only after reviewing it. The product validates the old definition against current references, creates a new current revision, restores its priority and leaves it paused. It does not erase newer history or restore its old enabled state. Review it in the editor and simulator before enabling. An archived record separately offers **Restore current archive as a draft**, also paused; restoring needs room among the 100 working rules.

If the record changed meanwhile, open the current saved version before choosing again. If the snapshot is no longer retained, it cannot be restored from this screen. Retained Free Gifts & BOGO rewards remain read-only here too. Use **Back to rules** to return.

Export configuration and import reviewed drafts

Use **Import / export → Export configuration**, or the export button under **Settings & integrations → Keep your configuration**. The downloaded `affstackdev-dynamic-pricing-export.json` includes current rule definitions, UUIDs, revisions, archive flags and settings/provider expectations. It does not include rolling previous revisions, Woo orders or order evidence. A database backup is needed to preserve those; this JSON is not a complete database restore.

To import:

1. Paste the export into **Export JSON**. With the enhanced interface, **Choose a JSON file to fill the preview form** can fill the same field. This alone does not submit or change rules.
2. Use a batch of 1–100 rules and at most 512 KiB (524,288 bytes). Split larger exports into smaller rule batches while keeping a valid export envelope.
3. Choose **When a UUID already exists: Keep the local rule, Replace and pause the local rule after review, or Create copies with new UUIDs**.
4. Choose **Validate and preview**. Read every action/message, local product/term name and normalized definition. The preview lasts 20 minutes and belongs to your signed-in account.
5. Correct every preview error and preview again. When valid, check **I reviewed the selections and any existing rules that will be replaced and paused**.
6. Choose **Apply this preview as drafts**. Check every row result, then review the saved drafts in Rules and Simulator. Enable them separately when appropriate.

Product and term IDs are local to each site. Import checks that references exist, but the same ID can identify a different item on another store. There is no automatic SKU/term remapping. Inspect the normalized definition for roles and history-product conditions too; these are not all listed in the short product/term-name summary. The imported currency must match the destination store; there is no currency conversion.

Keep local skips a matching UUID and creates missing ones. Replace uses the previewed revision of the matching working rule and pauses it; its former revision is retained locally. Copy gives each eligible input a new UUID. Archived input records and retained reward rules are always skipped. A matching local archive is skipped except when a nonarchived input is explicitly copied into a new draft. Restore archives through normal rule management, not by relying on import.

Import never overwrites local global settings, provider expectations or revision history with those from the file. It does not retain source numeric IDs or timestamps. Every imported or replaced rule is paused. Each row saves independently: a later error does not undo earlier successful drafts. A concurrent edit is reported instead of overwritten. Once processed, the preview shows results and no longer offers Apply, even if some rows failed.

If the connection fails during apply, inspect the original plan and saved rules first. Its original operation identity can recognize an unchanged completed save, but a new copy preview generates new identities and can create additional copies. Recorded failed rows are not rerun by that same plan; after correcting their cause, create a fresh preview for the intended remaining work. An expired preview must be recreated and reviewed. Never assume a missing response means no rows were saved.

Saved records and concurrent changes

Dynamic Pricing keeps rule definitions, revision history and global settings in its own tables, and pricing evidence with Woo orders. New clean defaults do not replace existing settings when the plugin is reactivated or upgraded. Previously observed provider expectations are retained separately and cannot be cleared by the settings form or import.

A rule save names the revision you opened. If another session saved first, your old input cannot overwrite it silently. Open the current saved version, compare the intended change and save from current information. First creation has a separate UUID identity, so the same unchanged first-save request can recover its original row. That does not promise that every later save can be repeated without a revision conflict. Bulk/import operations retain successful records independently, and an uncertain storage result requires checking the actual record before retrying.

Deactivation or uninstall preserves the product's records. Export provides a readable current configuration; it does not replace a database backup of all retained history and orders.

Read pricing information on a Woo order line

On a supported Woo order item, **Pricing offer:** shows the recorded rule name, revision, base → final price and currency per unit. **Inspect pricing decisions and refunds** opens the Order inspector for that order. An allocation note explains average unit pricing for a grouped/graduated offer and Woo's actual refund basis. A combination note explains that allowed coupons or credit followed the recorded pricing offer.

This compact line note displays saved evidence. It is not a recalculation with today's settings or proof that unchanged-looking monetary evidence remains verified. Use [Order inspector](#) and [Reports](#) for their detailed states, current charges and actual refunds. Turning off cart labels does not delete these order records.

Understand pricing explanations

A rule can be saved and enabled without being applied. The workspace explains the difference. **Rule is a draft**, **Currency does not match**, **Schedule has not started**, **Schedule ended**, **Customer role does not match**, or a

minimum/condition message means the corresponding rule requirement was not met. **Matched; another rule won** means it qualified but lost the configured overlap comparison. **No complete discount fits the quantity, floor or limits** points to the offer's quantity/caps/floor rather than a missing save.

Audience, B2B, Core, Bridge, credit and coupon messages identify the owner or rule that withheld pricing. Restore a missing connection or correct the stated configuration; do not change a shopper into an eligible consumer merely to make a sample discount appear. Free Gifts & BOGO/protected/foreign-priced lines remain owned by their existing product. **Another price writer changed the calculated price** requires resolving that competing integration. **Pricing decision unavailable** is the fallback when a reason is not recognized; investigate the specific saved/current context rather than treating it as a zero-price offer. The complete exact reason lookup is in the technical merchant reference.

Recover from validation or request errors

Message or situation	What to do
Permission or license required	Use an account with WooCommerce-management permission; resolve product access through Manage license. Retained inspection/export remain separate.
Enhanced navigation is disabled	Reload and use the standard forms.
Submitted form/sample/import is too large	Reduce the data to the documented bounds; split imports into valid smaller batches.
Invalid name, amount, selection, quantity, schedule or condition	Correct the corresponding field using the editor tables. Enter decimal digits without currency symbols/separators; compare schedule to the displayed site timezone.
Selected/referenced product, term or role no longer exists	Refresh current choices and replace/remove the reference. Cross-site IDs are not remapped automatically.
Condition group is full or condition changed	Simplify the tree or reload the current form and repeat the intended edit. Save is still required afterward.
Record changed in another session	Keep your input, open the current saved version in another tab, compare and reapply against its current revision.
Save could not be confirmed or response uncertain	Inspect the actual saved record before submitting again. Successful writes are not undone by a response failure.
Storage unavailable / additive update needed	Resolve database/setup availability; the stated recovery is to reactivate the plugin to retry setup. Existing records are preserved.
Rule limit reached	Archive an unused ordinary working rule, or review total retained capacity. Archives do not reduce the total retained count.
Archived rule	Restore it as a draft before editing, reprioritizing or enabling.
Reward rule belongs to Free Gifts & BOGO	Read/export the retained definition and configure the supported campaign in its owning product. It cannot be rewritten here.
Revision no longer retained	Choose an available snapshot; this screen retains only the latest 50 former revisions.

Message or situation	What to do
Import errors or missing confirmation	Correct every preview error, inspect local selections, then explicitly check the review confirmation.
Import preview expired or invalid	Create a fresh preview and review it again. Check for already saved rows before creating copies.
Some import/bulk rows failed	Read each result. Successful rows remain saved; address only the intended failed work.
History cannot be confirmed	Choose a real customer in account mode, repair inaccessible records or narrow lookback demands; unavailable is not zero history.
Screen could not load safely	Review saved configuration and plugin setup with an administrator; a generic rendering failure is not evidence that records were deleted.

The technical error reference supplies exact messages and service identifiers for support. Include the rule UUID/revision, operation and visible error when asking for help, without supplying license secrets or unrelated customer records.

What the pricing engine changes

Dynamic Pricing reduces the catalog prices of eligible purchased products. It supports bulk prices, graduated quantity breaks and complete groups sold for a set price. The saved rule determines who qualifies, which products count, when the offer runs and how far the price can fall. WooCommerce then calculates the order's coupons, tax, shipping and payment totals.

It does not add gifts, create coupons or redeem Store Credit itself. BOGO, Buy X/Get X or Y, cheapest/highest reward products and every-Nth rewards belong to Free Gifts & BOGO. An older reward rule can remain visible here for reference and export, but Dynamic Pricing keeps it paused and does not transfer or execute it automatically. See [checkout behavior](#) for how live prices are applied and removed.

Each purchased cart line has at most one selected pricing rule. Competing discounts calculate from the original catalog price rather than stacking on top of one another. Both the rule and live pricing must be enabled for normal storefront use; a saved draft can be included in a preview without enabling it.

What can enter a calculation

Pricing requires a confirmed eligible consumer. B2B accounts, unknown eligibility and unavailable required protection do not qualify, including B2B customers whose business discount is zero. Selecting a role or including drafts does not override this boundary. An unapproved coupon also holds pricing; selected allowed coupon combinations follow the separate [coupon policy](#).

The calculator accepts at most 100 rules, 200 cart lines and 10,000 units in total, including protected lines. Each line has a whole-number quantity from 1 to 10,000, and the store's price precision must be between zero and six decimal places. It requires a 64-bit environment. Inputs outside those boundaries cause a withholding reason rather than a partial authorized offer.

The result distinguishes matching candidates from selected rules. "Matching quantity" is the eligible product quantity considered for that rule. "Selected lines" counts cart lines the rule finally owns, not the number of

individual units reduced. A rule can match yet lose to another offer. Read the final selected lines after reviewing the conflict events, because an offer that wins an early conflict can be displaced later.

Entering amounts and understanding rounded prices

Enter amounts as plain unsigned decimal numbers, with up to seven digits before the point and six after it. The largest accepted amount is 9,999,999.999999. Do not use a currency symbol, thousands separator, spaces, minus sign, scientific notation or a trailing decimal point. Leading zeroes still count toward the seven-digit limit. Extra decimal places are rejected; they are not silently rounded when saving.

Percentages accept up to two decimal places and must be between 0 and 100. A base zero percentage or fixed reduction is allowed for graduated offers, so early units can remain at full price. Ordinary bulk percentage/fixed-reduction offers require a positive base value. A fixed-price value of zero is allowed. Quantity-break values can be zero, including on bulk rules.

The engine calculates internally to one millionth of a currency unit. Normal per-unit prices are then rounded to the store's price precision, with exact halves rounded upward. The Minimum unit price is rounded upward to that precision so the pricing rule does not go below it. At two decimals, a floor entered as 18.001 becomes 18.01.

For example, 10% off a 19.99 product calculates 17.991 before price rounding, giving 17.99 at two decimals. Adding the 18.001 floor makes the offer price 18.01. A fixed price of 25 cannot raise a product currently priced at 20; it simply produces no reduction. A zero fixed price with a zero floor can make an eligible purchased item free, but it does not add a separate gift.

Graduated/group offers and capped discounts can assign different intended prices to units on one cart line. WooCommerce receives a single average price for that line, calculated to six decimals and rounded upward by at most a few millionths across the line. A cap can also use only part of a unit's normal reduction. The examples below show this explicitly. WooCommerce remains responsible for the final line, coupon and tax rounding.

The rule's floor applies before permitted coupons. It is not a guaranteed final payable price after those later deductions.

Common fields when creating a rule

Open [Rules → Create a rule](#). A new rule starts disabled, with Percentage discount, Value 10, all eligible purchased products, sale products excluded, quantities counted across matching products, minimum quantity 1, no minimum spend or price floor, no dates and priority 100. Saving through the current editor uses the current rule format; older saved rules keep their supported calculations until edited.

Field	How to use it and its limits
Rule name	Required, up to 190 bytes after HTML is removed. It can appear in customer offer labels, so use a clear customer-facing name. Enter meaningful text that remains nonempty after WordPress sanitization.
Adjustment and Value	Percentage discount, fixed reduction per unit or fixed unit price. For a set-price method, choose Fixed unit price but treat Value as the total for the complete group. Amount and percent formats follow the preceding section.

Field	How to use it and its limits
Currency	Defaults to current store currency. The saved code is uppercased and must contain 3–8 letters. The rule applies only when it exactly matches the store currency. With a supported currency switcher its amounts are converted for shoppers in another currency (Shoppers in another currency).
Apply to	All eligible purchased products, selected products/variations, categories, tags, brands or global attribute values. Every selected scope except All needs at least one target.
Product and term lists	At most 100 submitted IDs per target or exclusion list, even if some are duplicates. IDs are positive whole numbers up to 2,147,483,647. Duplicate IDs are removed. Save checks whether the selected products and terms still exist.
Exclude products/variations	A parent product exclusion also excludes its variations; an individual variation can be excluded separately. Exclusion takes priority over inclusion.
Exclude products currently on sale	Checked on a new rule. It uses the product's current sale state. A definition supplied without this field outside the editor defaults to unchecked.
Consumer roles	Empty means any eligible consumer. Select up to 30 role entries; any selected role match qualifies. Role names use letters, digits, underscore or hyphen, 1–80 characters. Roles narrow the offer and never turn a business or unknown account into an eligible consumer.
Count quantities	Across eligible matching products combines their quantities. Within each cart line evaluates quantity/group pools separately for each line. The matching-spend condition still uses all matching lines.
Minimum quantity	Whole number 1–10,000; default 1. In line mode, individual lines below this quantity receive no offer.
Minimum matching spend	Default zero. Uses original catalog merchandise prices for matching eligible products, before this discount, coupons or customer tax calculations. Even in line mode, it totals all matching lines, including those later below the line's quantity minimum.
Minimum unit price	Default zero. Prevents this rule reducing a unit below the upward-rounded floor; does not raise a lower existing price.
Priority	0–9999; lower numbers are considered first. Equal priorities use the saved rule identifier for a stable order. Priority also decides ties under Compare offer savings.
Quantity breaks	Up to 20 distinct thresholds, entered one per line as 5 = 10. Quantity is 1–10,000; the value uses the rule's adjustment type. Blank lines are ignored, surrounding spaces trimmed, and breaks are sorted by quantity. Duplicate quantities reject.
Starts and Ends	Blank means no date limit. Dates use the WordPress site timezone; start is inclusive, end exclusive. End must be after start when both are set. Supported stored timestamps run from 0 through 4,102,444,800. Invalid date text is rejected.

Hold Ctrl or Command to choose multiple list entries. Product search accepts name or exact SKU and preserves unsaved form selections. Choose the matching Apply to option; selecting entries in another scope's list does not make those entries targets. Category targeting includes descendant categories through catalog facts. For global attributes, choose the attribute, refresh its choices without saving, and select the terms. A variation matches its selected attribute value, not every value available on its parent.

Saving also checks selected/excluded products, targeted taxonomies/terms and excluded category/tag terms. If an item has been removed, update the selection and save again. These IDs are local to the current site; importing another site's IDs does not establish the same products.

How older rule tiers work

Older rules support all products, product selections and category selections with the same three adjustment types. They use one winning quantity break for every unit on a line. With Count quantities across matching products, all those lines use the combined quantity; within each line, each line uses its own quantity.

The highest qualifying threshold replaces the base value. Below the first threshold the base value applies. The engine does not choose the most generous value: if a later threshold has a smaller discount or zero, that is the configured result.

With Priority, the first qualifying reducing rule owns the line. With Compare offer savings, a later rule wins only if it gives a lower unit price. Equal prices keep the earlier priority/identifier order. When old and current-format rules coexist, the complete-offer conflict rules described below apply to their competition.

Bulk, graduated and set-price methods

The current editor adds four executing Offer methods under How the offer applies. It also adds tag, brand and global-attribute targeting, category/tag exclusions, optional caps and additional conditions. All ordinary eligibility, minimums, dates and exclusions still apply.

Method or control	Meaning and limit
Bulk	Every unit in a qualifying quantity pool uses the highest qualifying break. The base value applies if no break qualifies. This is the default method.
Graduated	Each new rate starts at its numbered unit. Earlier units keep earlier rates; the base value applies before the first break. A zero starting percentage/fixed reduction is supported.
Mix and match: complete group for a set price	Consecutive complete groups each receive the configured total Value. Choose Fixed unit price as Adjustment. Do not enter quantity breaks. Incomplete groups stay at their original prices.
Order discount: an amount or a percentage off the matching products together	One discount over all matching lines: a percentage of their catalog total, or the Value once per cart with Fixed reduction. Fixed unit price, quantity breaks, group caps and unit caps are refused. See An amount off the order .
Units per qualifying group	Default 3; whole number 2–10,000. Used by set price. It must remain valid even when another method is selected.
Maximum groups per cart	Default 0 means no additional cap; maximum 10,000. Only a grouped method can use a nonzero value. Bulk/graduated reject a nonzero group cap.
Maximum discounted units per cart	Default 0 means no additional cap; otherwise 1–10,000. Counts units actually reduced for bulk/graduated; a set consumes every unit in its accepted complete group.
Maximum rule discount per cart	Default 0 means no additional cap; same money format as other amounts. Bulk/graduated can use the final partial amount on a unit. A set must fit its entire required saving in the remaining budget. An order discount is capped at this amount (rounded down to the store's smallest unit).

Method or control	Meaning and limit
Tags, brands and attributes	Select term IDs for the chosen scope. Attributes must be registered global product attributes; their stored taxonomy starts pa_ followed by 1-60 lowercase letters/digits/underscore/hyphen.
Exclude categories/tags	Empty by default; each list has the 100-ID limit. Any matching exclusion prevents the product receiving the offer.

An amount off the order

Choose **Order discount: an amount or a percentage off the matching products together** as the Offer method for offers such as "\$10 off orders of \$100 or more" or "15% off from \$50, at most \$12".

Control	Use for an order discount
Adjustment and Value	Percentage discount: that percentage of the matching products' catalog total, rounded half up to the store's smallest unit. Fixed reduction: the Value once per cart, not per piece. Fixed unit price is refused (order_requires_reduction).
Minimum matching spend	The amount the matching products must reach together before the discount applies. Eligible catalog prices count, before the offer, coupons and Store Credit.
Apply to, exclusions, roles, schedule, conditions	As for any rule: they decide which cart lines count towards the spend and share the discount. All eligible purchased products makes it a discount on the whole order.
Count quantities and Minimum quantity	Across matching products, the matching pieces must reach the minimum; within each cart line, lines below it are left out.
Minimum unit price	No piece goes below it. If the whole discount does not fit above the minimum unit prices, the order discount does not apply at all.
Maximum rule discount per cart	Caps a percentage discount.
Quantity breaks, Maximum groups per cart, Maximum discounted units per cart	Not used: leave them empty and 0 (order_offer_cannot_use_tiers, order_offer_caps). For a ladder, make one rule per step; with Compare offer savings the larger saving wins.

The discount is shared out over the matching lines in proportion to how far each line can go down (its catalog total above the minimum unit prices), in the store's smallest unit; any unit left over goes to the lines in product ID, variation ID and cart-line order. Each line's share is therefore a whole amount and the line totals together are exactly the discount lower; each line gets one unit price, its new total divided by its quantity at micro precision, as for set prices. Coupons, Store Credit, tax and refunds then work on these lowered line prices: a refund of one line gives back that line's recorded amount, its share of the discount included.

An order discount is one complete offer over all the lines it lowers: where another rule also fits one of them, the two compete as described in [Choosing between overlapping offers](#), and the order discount wins or loses all its lines together. Gift lines from Free Gifts & BOGO and lines another product owns are never part of it.

The order line's pricing record names the rule as for any offer; its allocation record adds the method order, the whole order discount and this line's share.

Older retained data can also contain “selected” rewards and cheapest/highest/every-Nth selection fields. Those are not current controls for choosing which bulk or set-price units are reduced. Active offers always use product ID, variation ID and cart-line-key order. For compatibility, saved retained selection values must still be one of those three, and the retained reward count defaults to 1, must be below group size, and must be exactly 1 for every-Nth. Invalid retained fields can prevent saving a definition even when an active pricing method does not use them. The technical reference describes these integration fields in its Schema 2 chapter.

Adding conditions and handling missing history

Use [Additional conditions](#) to add conditions inside ALL or ANY groups. ALL needs every condition to pass; ANY needs one to pass. Add/remove buttons update the form only; save the rule to apply the changed tree. Blank condition rows and empty groups are ignored. The complete tree allows 32 nonempty nodes, at most 12 submitted children in a group and a maximum of three levels below the root; groups and blank-field rows count toward the node bound.

Condition	Value and supported comparisons
Eligible cart quantity	Whole number 0–10,000; equals, does not equal, at least, at most, more than or less than. Counts all eligible lines, including zero-price ones and products outside this rule's selections.
Eligible cart catalog amount	Plain amount in the supported money range; the same six comparisons. Uses original eligible merchandise prices.
Matching quantity	Whole number 0–10,000; the same six comparisons. Counts positive-priced products after this rule's selection and exclusions, before its per-line quantity minimum.
Matching catalog amount	Plain amount; the same six comparisons, on original prices of those matching products.
Customer role	Role slug, 1–80 allowed characters; Equals/includes or Does not equal/excludes only.
Customer type	Enter guest or registered; Equals or Does not equal only. A logged-in account still needs eligible consumer status.
Retained paid order count	Whole number 0–10,000; six numeric comparisons, using the selected history window.
Net paid order amount	Plain amount; six numeric comparisons, using the selected history window.
Previously purchased product/variation	Positive product/variation ID up to 2,147,483,647; Equals/includes or Does not equal/excludes only.

History lookahead is 0–3650 days; zero means all retained history. It has no effect on non-history conditions. The [history chapter](#) defines which paid orders remain eligible, how refunds/currency are treated, and when bounded reads cannot confirm history.

Unknown history is not zero orders and is not proof that a customer has never bought something. Guests and hypothetical scenarios do not supply a real account's history. If an ALL group has a false condition, it fails; otherwise an unknown required condition leaves it unknown. If an ANY group has a true condition, it passes

even if another branch is unknown; otherwise unknown remains unknown. Only an overall confirmed true result grants the offer.

For example, ANY of “registered customer” or “at least two retained paid orders” passes for a confirmed registered consumer even if history is unavailable. ALL of those conditions does not pass until both are confirmed. Conditions use original quantities and amounts, so a competing discount never changes whether the spend condition qualified.

How units, limits and average prices are allocated

Matching units are processed by product ID, variation ID and cart-line key, then by unit position on that line. This determines graduated bands, group membership and which units use the remaining cap first. It is not cheapest-first or highest-price-first.

Across matching products, all qualifying units form one pool. Within each cart line, each qualifying line has its own pool; lines below the minimum quantity are left out. Graduated positions and set groups restart for each pool. The maximum groups, maximum discounted units and maximum discount amount remain shared across the entire rule, so an earlier pool can consume the cap before a later line.

Bulk/graduated caps count only units that receive a reduction. When less budget remains than a unit's normal reduction, that unit can receive the smaller remaining reduction. Set offers must fit a complete group: enough units, room within the unit/group caps, the required total discount within the budget, and enough price above each unit's floor. A group that cannot fit is skipped; a later group can still be accepted if it fits. A set total at or above the group's original total never raises prices.

For unequal set prices, the saving is allocated proportionally to each unit's available reduction above its floor. Any leftover millionths are assigned in stable unit order. A high floor can prevent a group altogether. This allocation determines intended unit buckets, but the cart keeps the original lines and quantities.

Every line receives one average unit price, including any unchanged leftover units. That average is rounded upward to a millionth, so the line may cost up to one less than its quantity in extra millionths compared with the intended buckets. A line that ends with no actual reduction is left unpriced by that offer. WooCommerce refunds use the recorded line amount, including later coupon/tax allocation; they do not select a special “discounted third unit” from the internal buckets.

The retained allocation records the method, buckets and quantities, number of discounted units, total accepted groups, intended line amount, charged line amount and the small averaging difference. Group count is the rule's overall count, so the same count can appear on more than one affected line. A recorded highest tier does not mean every unit on that line received it.

Examples you can reproduce in Simulator

Use an eligible consumer scenario, no coupon, store precision two decimals, and products with the stated catalog prices. Save the rule first and include drafts if it is disabled. The results below are merchandise amounts before customer tax and payment totals.

Example	Result
Four units at 10; 10% base, break 3 = 20; Bulk	All four receive 20%: average 8, total 32, saving 8.
Same configuration; Graduated	First two cost 9, next two cost 8: average 8.50, total 34, saving 6.

Example	Result
Three units at 10; Graduated base 0%, break 3 = 50	Two intended units at 10 and one at 5. Applied average 8.333334; charged catalog line 25.000002; engine saving 4.999998. The screen rounds money for display.
Three units at 10; Bulk 20%; maximum discounted units 2	Two intended units at 8 and one at 10. Average 8.666667; charged line 26.000001; saving 3.999999.
Three units at 10; Bulk 20%; maximum rule discount 3	First unit gets 2 off, second gets the remaining 1 off, third unchanged: average 9 and total 27.
Group of one unit each at 10, 20 and 30; group size 3, Value 48, floor zero	Saving 12 spreads as 2, 4 and 6: final lines 8, 16 and 24.
Same group with a minimum unit price of 8	Remaining reduction capacities differ: targets 9.333333, 16 and 22.666667, totaling 48. None goes below its floor.
Six units at 10; groups of 3 for 24; maximum discounted units 4	One complete group qualifies; no partial second group. Intended total 54 and average 9, with three units reduced.
Three units at 10; group of 3 for 24; maximum rule discount 5	No group fits because its required saving is 6. Unlike a bulk offer, the set does not apply only part of the discount.

When multiple products qualify, the ID order can change which product falls in a later graduated band or group. Preview the actual product combination, rather than assuming the most expensive products receive later discounts.

Choosing between overlapping offers

In Settings & integrations, When rules overlap offers Priority or Compare offer savings. Both consider lower priority numbers first and use the saved rule identifier to settle equal-priority order. Neither stacks the discounts of these competing rules on the same cart line; rules set to add to other offers come on top afterwards ([Rules that add to other offers](#)).

Ordinary uncapped bulk offers can compete separately on each line. Graduated offers, set-price offers and bulk offers with a maximum unit/discount cap compete as a complete allocation. Their plan includes all qualifying lines used in the allocation, even a line whose own units end up unchanged. This keeps groups, graduated ordering and rule-wide caps together.

Priority retains any earlier conflicting offer. Compare offer savings lets a new offer replace all conflicting offers only if its total saving is greater than the combined full saving of those offers. Equal savings keep the earlier offer. Replacing a complete offer removes all its affected lines together; a rule that lost earlier is not automatically reconsidered later.

For example, suppose a group offer on A and B saves 10 altogether. A later offer on A saving 9 loses under Compare offer savings, even if A alone would become cheaper. If the later offer saves 11, it replaces the complete group: B returns to its original price unless another later offer qualifies for B. A later saving of exactly 10 ties and loses. In Priority, the earlier group remains even against the 11 saving.

This comparison does not search every possible combination for a mathematically maximum cart discount. Use the simulator to review actual overlaps. Conflict events appear in evaluation order and can include a winner

later displaced by another offer. The first 200 events are shown; final prices and selected-line counts still cover the complete bounded calculation when the event list is truncated.

The calculator also produces a technical fingerprint of its final pricing lines, rule details, allocation, reasons and savings. That identifies a decision's contents; it does not authorize reusing old prices after eligibility or cart facts change.

Rules that add to other offers

Adds to other offers (How the offer applies) is off for every rule unless you tick it. A ticked rule does not take part in the competition above. After the competing offers are chosen, the ticked rules apply one after another in priority order (lower number first, then the saved identifier), each to the unit prices the earlier offers left: a percentage of that price, an amount off it, a fixed price below it, a set price or an order discount over the lines as they now cost. Each keeps its own Minimum unit price (it never lowers a line below it, and a line already at or below it is left as it is) and its own caps. A line no competing rule priced receives the ticked rule alone. Matching, Minimum quantity, Minimum matching spend and Additional conditions use the catalog facts, as for every rule.

The cart line's offer name lists the rules in the order they applied, joined by "+". The order line's pricing record keeps the first rule as its rule and the later ones, with the unit price before and after each, in their order; the order screen shows each ("Added to it: ..."), the Order inspector's decision lists them, and the evidence report shows such lines under all their rules together. The Simulator applies the same order. A rule saved before this setting existed has no such setting stored and keeps competing.

Running and interpreting a sample cart

Simulator previews saved rules without changing the live cart, products or rules. It requires merchant WooCommerce management permission and current pricing access. The [Simulator controls](#) provide product/customer search and retain your input between searches.

1. Find and select 1–20 simple products or individual variations. Select a variation rather than a variable parent. Enter one shared quantity from 1–500 for each selected product. Product count × quantity × saved rule count must stay at or below 200,000; one is used when there are no saved rules. All stored rows count toward that work limit, even disabled ones.
2. Choose Sample audience and roles for a hypothetical case, or Current customer eligibility and roles for account lookup. A sample starts as an eligible consumer/guest scenario with no roles. In account mode, choose a current-site customer or Guest; the service replaces hypothetical audience and roles with that account's current facts.
3. Include draft rules if you want disabled saved rules considered. This bypasses only the rule's enabled switch, not schedule, currency, consumer eligibility, conditions or retained reward ownership. Unsaved changes in a rule editor are not included.
4. Optionally enter coupon codes to check the combination policy, or mark that an additional unapproved coupon is applied. This checks whether pricing may combine; it does not calculate coupon amounts or approve Store Credit spending.
5. Run simulation and review Price explanation, per-product base/offer prices, Rule decisions and Conflict decisions. Compare final selected-line counts with matching quantities and open conflict details for the recorded savings comparison.

The simulator uses current catalog prices, sale flags, catalog selections, current store currency/precision, the current time and the saved overlap strategy. It can run while the live engine is paused. There are no custom date, price or currency override fields. Each sample product starts eligible; cart-specific gift flags, protected line metadata and other price writers are not replayed. Repeated identical product IDs supplied by an integration map to one result key rather than separate independent lines, although the submitted entries still count toward input limits.

In scenario mode, audience/type/roles are hypothetical. A logged-in sample type does not fetch a customer's purchase history. History conditions require account lookup and an available retained history window; an actual account can still have unavailable history because of read limits or unavailable records. Guest history remains unknown. A sample explicitly marked B2B or unknown does not receive consumer offers.

The sample merchandise reduction uses catalog-price arithmetic. It does not run gift/credit/other pricing engines or calculate customer tax, shipping, coupon deductions and final payment totals. An allowed coupon-policy result does not establish that WooCommerce applied a coupon or that credit is spendable. These boundaries explain why a preview can differ from an actual cart with more protections or instruments.

Results retain the settings revision, strategy, currency, audience result, history status and coupon-policy reasons alongside prices and rule traces. Account lookup shows eligibility and its decision owner; scenario results state that no customer eligibility was verified. Money display can round an advanced line's six-decimal average, so consult saved allocation evidence for very small differences.

Reading retained explanations

The inspector's snapshot is a bounded copy of an actual pricing decision, not another price calculation. It records product version, evaluation time, currency, catalog/tax basis, selection strategy, settings revision, rules/history identifiers, total saving, reasons, final line prices and selected rule identities. It keeps up to 30 reasons, 100 rule decisions, 200 conflicts and 200 lines.

Eligibility details include status, consumer/business classification, owning provider, revision, reason and validity period. The snapshot keeps the coupon policy's allowed/reasons summary and connection status. Its line summaries leave out full rule definitions and unit-bucket allocation; separate order evidence can retain those details. It does not contain the account's complete purchase history or coupon instrument list.

Rule outcomes mean: no matching lines shows the specific withholding reason; matched with no selected lines means competing offers won; partly selected means other offers won remaining lines; selected on all matching lines means all reducing candidates for that rule survived. This still counts lines rather than individually discounted units.

A separate snapshot fingerprint identifies retained content while ignoring the evaluation timestamp and the audience validity-expiry timestamp. It can stay the same when only those times change. It is not proof of present eligibility, a financial signature or the same fingerprint as the raw calculation. Use the [order inspector](#) for what reached checkout, and Simulator account lookup for current eligibility or an offer that never reached an order.

Why an offer was withheld and what to check

Result or problem	What to do
Invalid amount, percentage, quantity or selection	Use plain unsigned numbers, the documented decimal/range limits and valid product/term lists. Remove duplicate break quantities and unsupported field combinations. A blank field is not always a default when a value is required.
Set-price configuration rejected	Choose Fixed unit price, remove quantity breaks, use groups of at least two and review caps. Bulk/graduated must have a zero maximum-groups setting.
Condition error	Review tree size, group depth, allowed field/operator/value. Role, customer type and purchased-product conditions permit only equals or not-equals. Save again after correcting the form.
Invalid stored rule	A malformed saved row can withhold the whole evaluation. Review the stored definitions through supported administration; the engine does not quietly skip it.
Workload or input limit	Reduce the sample products/quantities or rule workload. For live integration errors, check the typed input and supported platform. A partial response is not approval to apply some lines.
Consumer ineligible or coupon present	Resolve the actual audience or coupon combination policy . Roles, draft previews and retail-priced B2B accounts do not bypass these checks.
Disabled, wrong currency, not started, ended or role mismatch	Check the corresponding saved field and current facts. Start includes its exact time; end excludes it. Draft preview bypasses disabled state only.
No eligible lines	Check product/variation scope, exclusions, sale status and protected or zero-price lines.
Minimum spend/quantity or no reduction	Check original matching merchandise, combined versus line quantities, rounded target and minimum unit price. A target at or above base produces no price change.
No complete discount fits quantity, floor or limits	Advanced allocation could not produce a reduction. Review complete group size, floor, caps and target together; this message does not identify one unique failed control.
Condition did not match	One or more required conditions are false. Review ALL/ANY logic and original matching facts.
Condition could not be confirmed	Required history/customer facts are unknown. Use account lookup where appropriate; unknown does not mean zero orders or no prior purchases.
Reward belongs to Free Gifts & BOGO	Retained selected rewards do not run here, including as drafts. Keep/export the definition and configure a supported campaign in the owner product.
Matched but another offer selected	Inspect final selected lines and conflict details. Check priority, stable equal-saving ties and the complete-offer comparison. Matching alone does not grant a price.
Simulator permission, license, storage or sample failure	Restore the stated access/storage prerequisite or correct the product/customer/quantity selection. A product with no valid catalog price cannot form a sample.

Technical readers can use the exact code and exception lookup in the technical reference.

Configuration/preview failures do not themselves enable a rule or mutate a cart, so correct the source input and rerun the preview before enabling the intended saved rule.

How pricing reaches checkout

Dynamic Pricing applies an eligible offer to purchased items' unit prices. WooCommerce then calculates tax, coupons, totals and payment. Customers keep using the normal cart and checkout, including supported WooCommerce cart and checkout blocks. The plugin does not turn an offer into a separate coupon or a delayed reward.

The engine starts paused. Its initial audience mode requires a verified consumer decision, ordinary coupon combinations start with no selected coupons, and Store Credit combinations start off. Configure these deliberately in [merchant settings](#). Pricing also requires current [license permission](#).

Recalculating and protecting the current total

Whenever WooCommerce recalculates the cart, Dynamic Pricing first removes the previous price changes it can still identify as its own, reads the current rules and customer eligibility, and evaluates the offers again. Quantity, product price, rule schedule, selected coupons and customer changes can therefore change the offer.

If another extension replaces a product or changes a price that Dynamic Pricing already wrote, the plugin does not overwrite that extension's work. It holds checkout for the affected row until the cart is refreshed and a safe price is established. Removing that row or restoring its ordinary product price can resolve the conflict after recalculation.

If license permission becomes unavailable, Dynamic Pricing stops applying offers. A cart restored from a session recalculates with current native prices. If permission disappears after an offer has already affected totals, the customer must refresh and review the current total before placing an order. Existing order history remains evidence of the order; it is not permission to apply the old offer again.

Which cart items can receive an offer

Ordinary simple products and variations can qualify. The rule's own product, category, quantity, schedule and other conditions still apply. See [the rule reference](#).

Dynamic Pricing excludes cart lines managed by Free Gifts & BOGO, unsupported product types, lines already priced by another extension, protected or related lines, and rows containing unsupported plugin-managed data. When Commerce Core or Commerce Bridge owns shared line information, their protection and ownership checks also apply. A store integration can explicitly exclude a line.

An excluded item normally remains in the cart at its existing price; exclusion does not delete it. If an offer does not appear, check the decision or Inspector before changing product prices or removing another extension's cart metadata.

Consumer eligibility

Consumer promotions depend on a current audience decision. Dynamic Pricing does not decide that a customer is B2B from their role or profile fields. A supplied B2B classification always excludes consumer pricing, even if another field claims that promotions are eligible.

In verified mode, a supported audience provider must supply the decision. In consumer-only mode, a store that has never required an audience owner can use its explicit merchant consumer-only policy. Once the plugin has observed an audience provider, B2B, Core or Bridge, it remembers that dependency. Removing the provider does not silently make everyone a consumer or remove protected-line requirements.

Fresh decisions and integration changes

Provider decisions must be current and valid for the requested purpose. Supplied expiry must be later than now and no more than five minutes ahead. The plugin checks fresh evidence again while applying prices and before accepting the order. A changed customer, owner decision, revision or shared context can require a new cart calculation.

Only one supported audience writer may provide the answer. Conflicting writers, unavailable owners, stale decisions or malformed integration data withhold the offer. Restore the intended provider and refresh the cart. A renewed expiry for the same decision does not by itself invalidate the offer.

Commerce Core and Commerce Bridge

Commerce Core supplies current audience context, stable cart-line identity and protection. Dynamic Pricing reads those facts before pricing and claims only its own pricing contribution on an eligible line. Core's shared records do not make Dynamic Pricing the owner of another product's offers or protected content.

Identity and ownership checks

When Core is available, its supported audience facade must be the single writer. The audience and the Core decision must agree about the customer and eligibility. Protected lines, relationships, commercial facts from another owner, or unsupported identity information prevent a new Dynamic Pricing offer on that row.

If Core was previously present and becomes unavailable, pricing is held until its supported interface is restored. A status card showing a connection is useful for diagnosis; the cart still performs current checks.

Stores using Commerce Bridge

Where Commerce Bridge is present, Dynamic Pricing also checks its identity, relationships, provenance and protection interface. A previously seen Bridge that disappears leaves a hold. When both Bridge and Core are available, both readers participate and Bridge supplies the line claim used by this pricing path.

Do not remove line identity fields to force an offer. Resolve the owning integration and recalculate the cart.

B2B connection and cleanup

The B2B connection requires compatible B2B and Commerce Core interfaces. The declared minimums are B2B 0.8.1 and Commerce Core 0.1.0, together with the required contract and capability checks. A version number alone is not sufficient if the loaded product does not expose the matching interface.

When B2B registers Dynamic Pricing, it receives a callback that can remove this plugin's current price changes when the customer is B2B or otherwise ineligible. Dynamic Pricing does not issue delayed order rewards, so that registration declares order awards excluded.

Failed registration withholds consumer pricing. Cleanup also holds when another writer has changed an owned price and safe removal cannot be confirmed. An eligible callback later in the same request does not erase that hold; refresh after resolving the cause.

Reading connection capabilities

The integration descriptor reports supported audience schemas, coupon combination policy, cart decisions, grouped conditions and purchased-unit allocation. Supported pricing methods are bulk, graduated and set price. Selected-unit rewards belong to Free Gifts & BOGO. The descriptor does not promise automatic transfer of rules to another product.

Retiring B2B before returning to consumer-only pricing

Uninstalling or deactivating B2B alone does not clear a remembered audience requirement. A deliberate return to merchant consumer-only pricing requires both accepted Core evidence that B2B's audience policy has been retired and the Dynamic Pricing setting explicitly changed to consumer-only.

The supported recovery path also requires B2B to be absent, no other audience-required condition, the sole Core facade, and the exact current Core missing-provider result. Dynamic Pricing then derives its own merchant consumer-only decision from the saved settings and accepted retirement record. It does not use the retirement record as the customer's audience classification.

If B2B returns, accepted evidence is revoked or changed, or the store switches back to verified mode, the next evaluation rechecks that choice. Retirement does not remove other line protections or coupon restrictions. The status report's accepted/not-accepted indication is informational and does not itself authorize pricing.

Combining offers with coupons and Store Credit

Coupon combinations are explicit. Dynamic Pricing first changes the eligible unit price, WooCommerce applies permitted ordinary coupons, and Store Credit applies afterward as payment. Each owner still validates its own instrument. Dynamic Pricing does not redeem credit or make an otherwise invalid coupon valid.

If an applied ordinary coupon is not an allowed combination, Dynamic Pricing withholds its offer for that calculation. Store Credit and gift cards never do. It does not silently remove the coupon. Review the new cart total whenever a coupon is added, removed or changed.

Selecting ordinary WooCommerce coupons

Enter permitted coupon codes one per line in [merchant settings](#). The selection accepts up to 50 unique codes, at most 200 bytes per code and 8,192 bytes for the full input. Blank lines are ignored and duplicate normalized codes count once. The saved selection uses coupon IDs, so its display reads the coupons' current codes; deleted coupons no longer appear.

Selected coupons must already exist, use a standard percentage, fixed-cart or fixed-product type, permit combinations, and contain no unsupported plugin-managed conditions. A coupon marked for individual use cannot combine. Native restrictions such as expiry, products or usage limits continue to apply.

Supported Store Credit

Store Credit and gift cards are payment, like a card or cash. Dynamic Pricing recognizes the owner's `afsd_store_credit` instrument through Store Credit's public `is_credit_coupon()` reader and never

withholds or changes a pricing offer because one is applied, including customer-referral reward credit. The offer price comes first, then allowed ordinary coupons, then the credit. Ownership, the usable balance—including any reservation for the same order—and the referral program's own restrictions stay with Store Credit. The Refer-a-Friend first-purchase discount is an ordinary-type coupon with plugin-managed conditions, so it still withholds pricing like other external coupons.

The **Allow recognized Store Credit with pricing offers** setting no longer changes pricing. The connection status still reports whether Store Credit's public interface (versions 1.2.0, 1.3.0 and 1.4.0, separate from its product release number) and validation hooks are available.

Changed combination terms

The plugin records a fingerprint of the selected policy and applied instruments in their application order. Changed coupon terms, metadata, selection or order require recalculation. Private codes and credit balances are not copied into the safe combination record.

Current credit balances remain the credit owner's responsibility. A recognized instrument is not a guarantee that it can fund the checkout; its owner still performs its fresh balance, ownership and redemption checks.

Shoppers in another currency

Rules are kept in the store's currency (WooCommerce's currency setting). When a shopper switches the shop to another currency with WPML Multilingual & Multicurrency for WooCommerce, or with FOX - Currency Switcher in its "multiple allowed" mode (the order is then paid in the shopper's currency), Dynamic Pricing uses every rule as a copy in that currency, at the switcher's own exchange rate in that request:

Part of a rule	In the shopper's currency
Percentage discounts and percentage breaks	Unchanged.
Fixed reduction, fixed unit price, set price, order discount amount and their breaks	Converted at the rate, then rounded to that currency's precision like any price.
Minimum matching spend, Minimum unit price, Maximum rule discount per cart	Converted.
Eligible cart catalog amount and Matching catalog amount conditions	Converted.
Net paid order amount (history)	Stays in the store currency; the history reader counts every order in the store currency, converting an order paid in another currency at its recorded rate.

The base price is the price the switcher charges for the product in that currency, also a price set for that currency in the switcher. With FOX, Dynamic Pricing's offer price is already in the shopper's currency, so FOX does not convert it again.

Each priced order line records the order's currency, the rule copy as it was used in that currency, and the conversion (the store currency, the rate and the switcher); the order screen shows "Rule amounts in USD were converted at 0.9 (WPML Multilingual & Multicurrency) into EUR for this order." Every order placed in another currency through a supported switcher also keeps the rate, which the history reader uses later. The order uses the rate of the checkout request, the same one its prices were calculated with.

Another currency switcher, FOX without "multiple allowed", or a currency the switcher has no rate for: the cart's prices are not the rules' currency and no offer applies, as without a switcher. In the store's own currency nothing changes.

Placing an order and reading its evidence

Before accepting a priced order, Dynamic Pricing checks that the same current rules, settings, currency, customer decision, line prices, quantities, coupon combination and relevant purchase history still support the result. Scheduled offers are checked again at the boundary. A rule that has ended or changed while the customer was checking out can require a refresh.

This applies to classic checkout and Store API checkout. The block checkout receives a conflict response if the pricing has changed, so the customer can refresh and review the total. Repeatedly submitting an old page does not preserve an expired offer.

When labels are enabled, an eligible cart row displays **Pricing offer** with the winning rule's name. The order keeps private evidence of its base/final unit prices, quantity, rule, audience and combination decision, plus optional allocation and financial evidence. These fields support the [order evidence and reports](#); they do not add a second charge or refund mechanism. WooCommerce continues to own order totals, payment, statuses and refunds.

Shopper display freshness

Shopper offer displays request fresh inputs without changing the cart price themselves. They stop offering a current projection when the engine or license is unavailable, audience or combinations are ineligible, a price conflict exists, or the supported display limit of 200 cart rows is exceeded. A display is an explanation of current inputs; the final checkout still rechecks the offer.

Continuity status and retained rules

The continuity report describes this plugin's current request. **Dynamic Pricing is paused.** means the engine is disabled. **Dynamic Pricing priced this cart.** means the current decision contains a saving. **Dynamic Pricing handles its own pricing rules.** means it remains the controller but no current saving is reported. **Pricing held:** is followed by the current reason.

The report does not run another cart calculation or authorize execution. It can expose a missing license, unreadable settings, failed B2B registration, missing Core/Bridge/audience interface, or a reason from the current decision. Diagnose that cause and recalculate rather than treating a status label as checkout approval.

Configuration that remains with Dynamic Pricing

Dynamic Pricing retains its configuration on uninstall. Its local configuration projection can describe a saved rule, policy, currency, tax basis and revision for an integration to inspect. It preserves the original definition and marks unsupported shapes for review.

There is no automatic successor, shared execution or transfer of pricing authority in this contract. Historical creator identity is not invented where it was not recorded. A retained Free Gifts & BOGO reward definition remains identified as a reward that Dynamic Pricing does not execute. Archived and disabled rules also remain distinguishable from enabled ones.

Checkout integration boundaries

The plugin connects to WooCommerce's normal session, totals, item-display, cart-hash and order-creation hooks. Classic and Store API checkout share the final validation. These local hooks are not a public remote control API and do not provide a separate customer permission to change pricing.

Extensions that alter product prices or cart-row data should use supported ownership and protection contracts. Otherwise Dynamic Pricing may exclude the row or require a fresh cart when two extensions change the same price.

Checkout messages and recovery

What you see	What to do
Your pricing offer has changed. Please refresh your cart and review the current total before placing the order.	Refresh the cart and review the recalculated total. Rules, eligibility, coupon terms or license permission may have changed.
A pricing offer could not be confirmed. Please refresh your cart and review the total before placing the order.	A price writer conflict remains. Refresh; ask the merchant to resolve the conflicting pricing integration if it persists.
Please refresh your cart to review its current pricing.	The order-item price or quantity differs from the evaluated cart. Recalculate before ordering.
B2B could not register Dynamic Pricing cleanup. Consumer pricing is withheld.	The merchant should restore the compatible B2B/Core connection.
A pricing offer could not be removed safely. Please refresh your cart and review the total.	B2B cleanup could not establish safe removal. Refresh after resolving any conflicting price writer.
A coupon is not selected, has individual-use rules, or has another plugin's conditions	Select a supported ordinary coupon in settings, or remove the incompatible coupon and recalculate.
Store Credit contract, validation or recognition is unavailable	Restore the supported owner integration; do not rely on the combination checkbox alone.
Audience provider missing, conflicting, stale or unavailable	Restore one supported provider. A remembered owner does not disappear when its plugin is deactivated.
Saved configuration needs review	Inspect the retained definition. The projection leaves its original data unchanged.

Interpreting integration diagnostics

The technical reference lists exact machine codes and interface messages. B2B compatibility diagnostics distinguish missing descriptors, identity mismatches, versions below the supported minimum, wrong contracts and missing capabilities. Correct the named dependency rather than changing customer data to force eligibility. For offer-selection explanations and rule-level reasons, use [engine diagnostics](#).

Choose shopper displays

Open **Dynamic Pricing → Settings & integrations → Shopper displays** to choose how customers see current offers. These displays explain eligible prices; the cart still confirms the final total. They require pricing to be

enabled and available under the current license. New installations start with all five display checkboxes off. Use the ordinary settings save control after choosing them; [merchant settings](#) explains saving and concurrent changes.

Control	How to use it
Show quantity offer examples on product pages	Adds a table of eligible quantity examples for the current product.
Product display position	Choose Before the add-to-cart form, After the add-to-cart form, or Shortcode placement only. The initial position is Before.
Show verified pricing savings in the cart	Shows the current pricing reduction when the cart's actual selected prices agree with it.
Show next quantity / spend target in the cart	Shows one supported extra-quantity target that can lower a current product's unit price.
Enable the current offers page shortcode	Allows a page that browses current simple-product offer examples.
Show the offer price in the shop and on product pages, with the old price crossed out	Shows a product's offer price where WooCommerce shows its price: the shop, category pages, product grids, the product page and a chosen variation. See Read offer prices in the shop and the cart .

For manual placement, use `[affstackdev_pricing_table]` on a product page, or `[affstackdev_pricing_table id="123"]` for a specific product ID. Product tables must still be enabled. Place `[affstackdev_pricing_summary]` beside a Cart block when you want the savings/target display there; savings or targets must be enabled. The classic cart places this summary automatically after its cart table. Place `[affstackdev_pricing_offers]` on an ordinary page for the offer browser, with its checkbox enabled. Shortcodes do not turn on their corresponding setting and do not change prices by themselves.

A theme or product template must actually render the relevant classic add-to-cart hooks for automatic product placement. Use the shortcode where automatic placement is unavailable. The offer displays do not replace the product's normal purchase controls.

Read quantity offer examples

A product card is headed **Quantity offer examples** and names the product. Its table shows **Quantity**, **Average each** and **Total** under **Current examples, before coupons and shipping**. Each row represents a cart containing only that product at that quantity. Adding other products, applying coupons, changing delivery details or changing eligibility can alter the result. The table follows the shop's tax-display choice and tells the shopper whether prices include applicable tax or exclude tax. Final totals are confirmed in the cart.

The display considers up to 12 selected quantity points, including minimum quantities, spending thresholds, group quantities, tier breaks and supported condition thresholds. It does not list every possible quantity. Only examples with a selected valid offer appear. Stock and sold-individually constraints are checked before a row is shown.

For a variable product, the initial message is **Choose product options to check quantity offers for that variation**. With the normal variation controls and JavaScript available, choosing a concrete variation refreshes its examples; resetting choices returns to the parent message. An unresolved “any” variation choice is not treated as a confirmed concrete example.

Examples are withheld for unavailable, unpublished, invisible, password-protected, unpurchasable or out-of-stock products, unsupported product types, nonpositive/unreadable prices, conflicting external price changes or exclusions from a connected product. A line already marked ineligible in the shopper's cart can also suppress its example. Missing examples do not grant a discount or demonstrate that a product has no possible rule under every other cart scenario. Check the current cart and, as a merchant, the [Simulator](#) when investigating eligibility.

Read offer prices in the shop and the cart

With **Show the offer price in the shop and on product pages, with the old price crossed out** on, a product whose price a rule lowers shows its normal price crossed out and the offer price, the way WooCommerce shows a sale. The price shown is what one more piece would cost in the shopper's current cart: the shopper's role and history, a cart amount already reached and pieces of the product already in the cart count. When no rule lowers that price, for example a rule that starts at 5 pieces, the normal price stays (the quantity examples show such offers). For a variable product the range stays as it is; choosing a variation shows that variation's price, and WooCommerce's variation price is then shown even when all variations cost the same.

The page's own HTML keeps WooCommerce's normal price inside a marker that is the same for every visitor, so page caches and caching plugins keep working. The storefront script then asks your store privately for the shopper's offer prices of the products on the page (as for the other displays: a short-lived answer for the actual session, never stored in the page or the browser) and shows them; whenever no offer applies or the request fails, the normal price is shown again. Without JavaScript the shopper sees the normal price; the cart applies the offer.

In the cart, a line a rule priced shows its normal price crossed out before the offer price: the block cart does it with WooCommerce's own sale display, the classic cart and the mini-cart through Dynamic Pricing.

B2B business accounts, a paused engine or license, a coupon that is not allowed with offers and products excluded by a connected plugin show no offer price, as in the cart.

Read current cart savings and targets

Pricing offer savings: appears only for a positive confirmed reduction whose selected unit prices match the actual cart products. It is before coupons and Store Credit, using the cart's tax-display preference. Do not subtract this message again from the cart total: the selected prices already contribute to that total, and allowed coupons/credit are separate deductions.

A next-target message reads **Add N more of PRODUCT to reach a lower unit price in this cart.** It is calculated against the current complete cart, including matching quantity/spend thresholds. The accompanying **Additional merchandise before offers:** amount shows the extra quantity at its native price, not a promise of the final increase in checkout total.

The display offers one target with the fewest additional units among its bounded candidates. It checks available stock, including other cart quantities sharing the same stock pool, and skips sold-individually products. It tries at most 24 supported projections and does not search every possible mixed-cart combination. No target means none was found within that supported search; it does not imply that every possible purchase has been compared. The message does not add products, change quantities or issue a free gift. The shopper makes any cart change and reviews the recalculated result.

Browse current offers

A page containing [affstackdev_pricing_offers] shows **Current pricing offers** for the current shopper. Each card contains the product's quantity examples and a **View product** link. **Previous** and **Next** navigate candidate pages.

Each page examines up to 12 eligible catalog candidates in ascending product ID order. Some candidates may have no qualifying offer, so a page can show fewer cards or **No pricing offers are currently available on this page**, even when a later page exists. The browser covers visible, published, in-stock simple products; select options on variable product pages to check their concrete offers. It does not list private products or assume that one visitor has another visitor's eligibility.

Fresh prices and the fallback page

The normal page contains generic offer placeholders and links. Current personalized content is requested privately for the actual session; it is not saved in the page's public HTML or persistent browser storage by this plugin. Each display refreshes when its short-lived response expires, normally within a minute and sooner at relevant audience or rule schedule boundaries.

The script also refreshes after supported cart/checkout changes, variation selection, browser focus and page visibility changes. It clears old personalized prices before asking again. A hidden tab stops displaying the previous personalized content until it becomes visible and refreshes. If the request fails, stale content stays removed and another request is attempted after approximately 30 seconds. A network request is allowed 10 seconds before it is aborted. These refreshes do not place an order or change the cart.

The fallback links remain usable without JavaScript: **View current quantity offers**, **View current cart savings and targets**, or **Browse current pricing offers**. They open a private **Current pricing offers** page with the relevant result, or **No pricing offer is available for this selection right now.**, and **Back to shop**. The fallback page is marked not to be indexed. It does not guess an offer when eligibility, storage or a connected provider is unavailable.

Tables wrap long text and amounts, adapt padding on narrow screens, and provide captions and keyboard focus outlines. If no display appears, check its checkbox, master pricing setting, active rule, current audience, stock, coupon compatibility and connected-product status. A public empty result deliberately does not expose merchant diagnostic details.

Inspect an order's pricing decisions

Open **Dynamic Pricing → Order inspector**. Under **Order decision inspector**, enter **Order ID** and select **Inspect order**. This screen requires WooCommerce management permission. It remains available as retained records when the pricing license is paused; that access does not authorize new pricing work. An unavailable order displays **This order is unavailable**.

The ordinary WooCommerce order also offers a compact pricing explanation and an **Inspect pricing decisions and refunds** link; order-line evidence explains those details. This inspector is for what was retained on an order. To ask why a customer currently does or does not qualify, or why an offer never reached checkout, use the Simulator and its account-lookup option instead.

Checkout pricing decision shows the recorded evaluation time, currency and tax basis. If there is no supported decision snapshot, the screen explains that older line evidence remains on the WooCommerce order and

missing history is not reconstructed. It does not rerun today's rules and pass that result off as the original decision.

Read rule and conflict decisions

Eligibility owner: identifies the retained eligibility source, classification and result. Hypothetical/unconfirmed input is labeled accordingly. Explanations can include exact reason codes for support.

Rule decisions lists **Rule / revision**, **Matching quantity**, **Selected lines** and **Outcome**. The outcome distinguishes a rule that did not qualify, one that matched but lost to another offer, a partial selection and a selection on all its matching lines. The saved revision matters: editing a rule today does not rewrite what that original revision meant.

Expand **Conflict decisions** to see the selected and competing offers and their catalog-basis reductions. Events appear in evaluation order; an early selection can be displaced later. Read the final rule table for the final selected counts. Compare-savings compares supported complete conflicts; it does not prove that every possible cart combination has been explored. If the event list was capped, the screen says only the first 200 conflicts are shown, while the final prices/counts still describe the calculation. **Line eligibility and selected owners** expands each retained line's owner or reason.

Read observations and current amounts

Current order amounts comes from the current Woo order and refund records. It remains current even if you selected an older decision through an observation link. The precise measures are described under [report values](#).

Retained observations lists checkout, status, refund and refund-deletion observations captured while Dynamic Pricing was active and licensed. Expand a timestamp/kind/status entry to read its structured details, or use **Inspect this observation's decision** to inspect the associated saved decision. The list deduplicates identical states and retains at most 100 observations per order. At that limit, older observations remain, new entries stop, and a message explains the cap. Current Woo figures still load above.

This is not a payment ledger or a complete activity log. It cannot fill gaps from before installation, disabled hooks or periods when pricing access was unavailable. The inspector does not edit an order, refund money, restore a deleted refund or regenerate an old decision. A recorded observation is not proof that a gateway payment succeeded.

Use the pricing evidence report

Open **Dynamic Pricing → Reports → Pricing evidence report**. Choose **Created from (UTC)**, **Created through (UTC)** and **Currency**, then select **Load report**. Defaults are the first day of the current UTC month through today and the current store currency. Dates must be real YYYY-MM-DD dates in order, covering at most 366 inclusive calendar dates. The entire ending UTC date is included.

The report selects orders by creation date, not completion date or refund date. It uses their current status and current refund records, including registered statuses beyond completed. Currencies are kept separate. Each page scans 25 orders, descending by order ID, and totals only that page. Some scanned orders contain no pricing evidence and do not appear as priced rows.

Page / pages / orders scanned: shows where you are. **This page only** labels the totals. The **Orders containing pricing evidence** table shows **Order**, **Status**, **Rule / revision / priced lines** and **Verified completed reduction**,

excluding tax. Select an order number to inspect it. **Previous page** and **Next page** retain the filters. Empty or sparse priced rows do not mean the date range contained no orders.

An order whose evidence cannot be read within the supported limits is explicitly omitted from totals and identified by order number. Supported per-order limits are 200 product lines, 100 refunds and 1,000 cumulative product/shipping/fee refund lines. Read that order's native records when its history is outside those limits. The report does not replace omitted amounts with zero and claim complete coverage.

Interpret each amount

Measure	Interpretation
Recorded checkout offers	The saved base-versus-final catalog reductions, separated by whether the catalog basis included tax. They are not product-page views, impressions or abandoned-cart estimates.
Verified reductions on orders with completion evidence, before coupons / excluding tax	Reduction supported by a matching financial line snapshot on an order with a completed date or current completed status.
Current priced-line totals after coupons, before refunds / excluding tax	Current Woo line amounts for lines carrying pricing evidence.
Actual priced-line refunds / excluding tax	Actual refund amounts linked to the original priced product lines.
Actual priced-line tax refunds	Tax refunded on those linked priced lines.
All refunds on these orders / including tax, shipping and other items	Full Woo refund amounts, including refunds not attributable to a pricing offer.
Order-level refund amount without line allocation	Refund value not allocated to a product, shipping or fee refund line.
Pricing lines with missing, changed or unverified financial snapshots	Lines for which the original pricing-to-order comparison cannot currently be verified.

The report records the offered reduction before allowed coupons and credit. It never subtracts refunds from that original offer reduction and does not infer how much of a refund should be assigned to a discount rule. WooCommerce refunds follow the order's actual line amounts, including later coupon and tax allocations. An average-unit-price offer does not acquire a different refund method from this report.

Completion evidence means a completed date exists or the current status is completed. An order later refunded or cancelled can retain its completed date and remain in that measure. That is why current status, original reduction and actual refunds are shown separately.

Changing a saved order item's product, quantity, subtotal, total, tax or tax class can make its financial snapshot no longer match. The order and original evidence remain; a changed line is not silently treated as a newly verified reduction. Deleting a Woo refund changes the live refund figures, while an earlier retained observation can still show that a refund existed then.

Evidence exceptions: can report unsupported line evidence, unusual refund signs or allocation mismatches. Review the authoritative Woo order/refund and the retained decision before drawing conclusions. The measures do not establish profit, incremental sales or causal sales uplift, and a page total is not a complete period total.

Preserve configuration and distinguish exports

There is no report CSV/JSON export button or observation-file download on these screens. Observation details are readable in the inspector. **Export configuration**, or **Export saved pricing configuration** when pricing is paused, downloads settings and rule records; it does not include customer orders, report amounts or retained order observations. Follow [Import / export](#) for configuration backup and controlled draft import.

Saved rules and order evidence are retained when pricing is paused. The screens do not erase an order's financial records. A missing observation is not a reason to delete the Woo order or fabricate a replacement decision.

Use a merchant assistant proposal

A compatible merchant assistant can read rule definitions, simulate a sample scenario and stage a single draft proposal. It requires your WooCommerce management permission and current Dynamic Pricing access. It cannot use this interface to read customer history or orders, look up an account, submit coupon codes, automatically save a rule or enable pricing rules. There is no public customer assistant access to this adapter.

Read requests return rules in pages of 25, including paused and archived records, with their current identities, revisions, priority and definitions. Sample simulation uses 1–20 selected simple products or individual variations and a common quantity from 1–500, subject to the shared work limit. It can model consumer/B2B/unknown audience, guest/registered customer type, roles, paused drafts and a blocking coupon scenario. These are hypothetical inputs, not confirmation of a real customer's entitlement. The local product catalog still supplies prices and selections. For account-based investigation, use the normal authorized [Simulator](#).

Open and review the proposal

A staged proposal returns a review link; staging has not saved a rule. Open that link as the same merchant account that created it. It lasts approximately 20 minutes and cannot be handed to another account as save permission. A new proposal gets a new identity. A replacement must target the exact current revision/UUID of a paused, unarchived rule. An enabled or changed rule must be handled through its current normal management state before proposing a replacement.

The link opens **Review assistant draft**. Read the proposal name, create/replace explanation, **Target ID / expected revision / priority**, and its resolved local product/term references. Product and term IDs belong to this store; a plausible name from an assistant is not enough to confirm a selection.

The summary shows **Pricing method**, **Adjustment**, **Selection**, **Minimum matching quantity**, **Minimum matching spend**, **Minimum unit price**, **Quantity breaks**, and **Additional conditions**. Expand **Complete normalized definition: exclusions, tiers, conditions and limits** to review the full definition, including exclusions and nested conditions. The short summary alone does not replace that review.

Save a paused draft deliberately

After reviewing the complete proposal, tick **I reviewed the definition and want to save this paused draft.**, then select **Save paused draft**. The checkbox begins unchecked. Saving rechecks access, the preview, local references and the target revision. A replacement preserves the earlier revision and leaves the rule paused.

The result appears on the review screen. **Open saved draft** opens a successfully saved rule for ordinary editing and review. Saving through this screen never enables it; use the normal rule-management controls when you have finished checking it.

If the link expired, storage failed or the rule changed, read the message and generate a fresh proposal from the current saved record. If a request was interrupted after saving, inspect the existing saved draft/result before creating another proposal. The reviewed operation retains an outcome to support safe retry; do not assume a missing browser response means no draft was saved. A failed stored result needs correction and, where necessary, a new preview.

Resolve display and evidence problems

Problem	What to do
Product or cart display is empty	Check the display checkbox, master pricing, current license, audience, stock, quantity, coupon policy and competing product ownership. Use the fallback link or the merchant Simulator.
A displayed example differs from a mixed cart	Compare the example's single-product assumption with actual quantities, other products, coupons and delivery/tax context. Review the current cart total.
No next target appears	The product may already be at its best applicable tier, sold individually, stock-limited or outside the bounded target search. A target is advisory and does not alter the cart.
Order decision is missing	Inspect any older line evidence on the Woo order. Use current simulation for current eligibility; original missing history cannot be reconstructed by this screen.
An order is omitted from a report	Open the specified order and review availability, numeric amounts and supported item/refund limits. The displayed page totals intentionally exclude the unreadable projection.
A line is unverified or has an evidence exception	Compare its current quantities, totals and tax data with the saved decision and native refund records. Do not replace original evidence to make a report look complete.
Date/currency validation fails	Select one supported store currency and valid UTC dates, ordered correctly and within 366 inclusive days.
An assistant request is rejected	Use a documented operation and complete local rule definition within the size/quantity limits. The adapter does not accept account IDs, order requests or coupon-code lookup.
Proposal cannot be saved	Confirm the full review, use the same merchant account, refresh an expired proposal and reconcile changed/active/archived target rules. Check the existing result before retrying an uncertain save.

Developers can use the technical error lookup for exact codes and state effects. These recovery paths preserve WooCommerce's order and refund authority.

Activate and manage the Dynamic Pricing license

Dynamic Pricing needs its own license key from your AffStackDev account. Activate it before using pricing operations. Suite is optional and needs no key of its own: it can organize the product's navigation, but its

presence does not replace the Dynamic Pricing license. AffStackDev's current plans do not require a Suite license.

The license screen remains available for recovery even when WooCommerce is unavailable or the pricing workspace cannot run. Open **Settings → Dynamic Pricing license** as a site administrator. The Dynamic Pricing workspace also provides a **Manage license** link. A merchant who can manage WooCommerce but cannot manage site settings can view retained pricing records; a site administrator must perform license activation and recovery.

The product saves your rules, settings and existing order evidence when licensing pauses. It does not delete them or change historical order totals. It does stop current pricing operations and protected plugin updates when the license is missing, restricted or past its permission deadline.

What remains available during a license hold

The main workspace changes to **Retained pricing records**. Merchants with the normal WooCommerce management permission can open **Order inspector**, **Reports**, and **Export saved pricing configuration**. The export preserves the saved rules and settings so they remain accessible while licensing is recovered. The rule editor, normal settings changes, simulator, import/review work and merchant assistant require current licensing permission.

New storefront offers, quantity tables, savings messages and progress displays are withheld. Dynamic Pricing stops publishing new price decisions and labels. Existing shopping sessions are recalculated without its unavailable offers. If permission disappears after an offer has already affected a total, checkout can ask the shopper to refresh the cart and review its current total before placing the order. This avoids submitting a stale offer. It does not block an ordinary purchase merely because an unused Dynamic Pricing rule is saved.

Already saved order evidence remains readable. New Dynamic Pricing observation entries for checkout, status changes or refunds are not recorded while licensing is held; WooCommerce still owns the order's payment, status and refund operations. Restoring permission does not retrospectively manufacture observation entries for events that occurred during the hold.

The server can restrict an installation immediately. Its stop, lock and read-only restrictions all pause Dynamic Pricing operations and protected updates; the read-only wording does not keep current pricing active. Recovery controls and retained records stay available. See the technical permission boundaries for integration behavior.

Read the license screen

The screen begins with a status label and explanation. **Open AffStackDev account** opens the account website in a new tab for your key, entitlement and activation management. Dynamic Pricing does not ask you to store an account password in WordPress. When the license is usable, **Back to Dynamic Pricing** returns to the product workspace.

Detail	What it tells you
Site	The website address to which this installation is bound.
Environment	The licensed environment: Production, Staging, Development or Local.
Installation	The final eight characters of the installation identifier, preceded by an ellipsis. This is a reference for the installation, not the secret credential.

Detail	What it tells you
Plan	The plan accepted by the licensing service, or a dash before acceptance.
Last accepted check	When the service last accepted this installation.
Last attempted check	When this site last attempted a license request. An attempt alone does not renew permission.
Next automatic check due	The next retry or routine check due time.
Current permission deadline	The earlier applicable limit on the currently accepted permission, when one is displayed.
Server-authorized deadline	The accepted term or grace deadline, when displayed.

Times are shown in UTC to the minute. A dash means no timestamp is available. Some denied or recovery states do not display computed deadlines; the current status message remains authoritative. A previous successful date does not mean the license is currently usable.

When a saved readable credential exists, the screen offers **Check now / recover activation**. If a remote release is pending, the same control is labeled **Retry pending deactivation**. These actions use the saved installation credential; you do not need to enter your full key for routine checks.

After an action, a temporary message appears once on the license page. Read the current status below it as well: a completed request can still leave a server restriction or an expired entitlement to resolve. The message disappears after it is displayed or after two minutes.

First activation or replacement

1. Open **Settings → Dynamic Pricing license** as a site administrator.
2. Use **Open AffStackDev account** to locate the complete Dynamic Pricing key for this product.
3. In **Activate license**, enter the key into **Dynamic Pricing license key**. The field is masked, starts empty, and accepts up to 128 characters in the browser. The server-side limit is 128 bytes, so use the complete key exactly as provided.
4. Choose **Licensed environment**: Production, Staging, Development or Local. The initial choice follows WordPress's environment setting. Select the appropriate licensed environment before binding the credential.
5. Select **Activate license** and read the returned status, site, environment, plan and accepted-check time.

Production activation can require an HTTPS site address. If the account has no available site slot, manage its existing activations in the AffStackDev account before retrying. The product does not automatically remove another site's activation to make space.

Dynamic Pricing uses the full key for activation and then stores an encrypted installation credential for subsequent checks and protected updates. It does not retain the full key in its settings or display the saved installation secret. Secure storage requires a supported encryption extension on the host. If the host cannot securely store the credential, activation stops; ask the host to restore encryption support before trying again.

With an active credential, the section is titled **Replace / recover license** and the environment selector is locked. A replacement uses a new protected candidate while retaining the older installation credential until the result is

resolved. If an activation response is interrupted, the screen warns that a previous activation needs recovery and disables another key submission. Use **Check now / recover activation** first. The service may already have accepted the candidate, so repeatedly entering keys is not the recovery procedure.

A failed replacement does not automatically erase a separately retained older accepted license. Its continued use still depends on current deadlines and server restrictions. Changing the licensed environment requires resolving or deactivating the current credential first. Changing WordPress's own environment after activation is treated as an identity change and uses the reconnection procedure below.

Recover interrupted checks and understand status

Dynamic Pricing normally checks its license every 12 hours, with a small spread around the scheduled time. A previously accepted license can remain usable for at most 72 hours after the last successful local acceptance, limited earlier by the server-authorized term or grace deadline. Merely visiting a page, scheduling a check or receiving an error does not restart that period. At the deadline itself permission stops.

A term grace period is granted by the licensing service. Dynamic Pricing does not invent extra term grace after an accepted active term expires. If a renewal or entitlement change has been made in the account, use **Check now / recover activation** to obtain the new accepted state.

Status label	What to do
Activation required	Activate this product's key. There is no unlicensed introductory operating period.
Activation recovery pending	Use Check now to resolve the saved candidate before submitting another key.
Validation required	A credential is saved but no accepted current validation is available. Use Check now.
Licensed	The installation has accepted permission within its applicable deadline.
Licensed — validation overdue	Permission is still within its deadline, but a routine check is overdue. Use Check now if automatic recovery has not completed.
Licensed — connection grace	The service could not be reached. Automatic retries continue within the existing deadline; restore connectivity and check again.
Licensed — term grace	The server granted a limited term grace period. Review the account before its deadline.
Licensed — term and connection grace	Both conditions apply; the earlier limit wins. Resolve the account and connectivity conditions.
Licensed — deactivation pending	A remote release was requested but not confirmed. Use Retry pending deactivation ; do not assume the slot is free yet.
License validation expired	The server deadline or 72-hour connection limit has been reached. Restore service/entitlement and Check now.
License restricted by server	Current pricing and protected updates are paused by a server restriction. Resolve it through the account/service, then Check now.
Activation missing	The service did not find this exact installation. Review its activation and recover the correct site binding.

Status label	What to do
License could not be authenticated	Check that the key/installation belongs to Dynamic Pricing and the correct account.
License unavailable	The entitlement is unavailable. The public response does not identify whether it expired, was suspended, was revoked or has another restriction; review the account.
License needs attention	The service denied the installation. Review its account/activation and validate again.
Local license state needs recovery	Cached state failed its integrity check, while the installation credential was retained. Use Check now.
System clock needs attention	The site's clock moved backward beyond the tolerated five minutes. Correct it and Check now.
Site identity changed	Follow explicit reconnection for a moved/cloned site or changed WordPress environment.
License credential needs recovery	The encrypted credential can no longer be read. Follow credential recovery below.
Site deactivated	The remote release was confirmed. Enter a valid Dynamic Pricing key to activate again.

The technical status reference defines exact state precedence and deadline calculations. Status labels describe the current installation; they are not evidence about another product, user or license account.

Automatic retries and safe browsing

The first availability failure retries after roughly 15 minutes, the second after about one hour, and later failures after six hours. A service throttle may increase the delay up to six hours. These retries do not extend accepted permission. An explicit **Check now** remains available for an administrator after the underlying problem is corrected.

Scheduled WordPress tasks are one recovery path. Safe administrator page visits can also retry overdue checks. Ordinary public browsing can queue a short check at the end of the request; hosts that support it finish sending the page before that check. The automatic network timeout is four seconds.

This browsing fallback excludes purchases, cart/checkout/account pages, account/order actions, API requests, background cron, command-line work, AJAX and other mutations. It does not ask shoppers for a license key. AffStackDev products share a one-check budget on such a visit, so another product's check can defer Dynamic Pricing until a later visit. A low-traffic site still needs working scheduled tasks or an explicit administrator check; an event being scheduled is not a guarantee that WordPress will run it on time.

If another license action is already in progress, wait briefly and retry. The product serializes activation, validation, deactivation and reconnection so they cannot overwrite one another's state. Avoid clearing saved credentials manually to bypass a pending operation; use the recovery controls that retain the original request identity.

Reconnect a moved site or unreadable credential

The license is bound to this installation's website address, WordPress environment and local cryptographic configuration. Moving a site, cloning it, changing its address or WordPress environment, or replacing WordPress security salts can make the old binding unusable. Changing the WordPress environment type (WP_ENVIRONMENT_TYPE), for example when a staging copy becomes the live site, counts as a changed environment: release this site's activation first while the identity still matches, then reconnect after the change and activate again, so the released place is free.

For **Site identity changed** or **License credential needs recovery**, the screen provides **Reconnect after a move or credential change**:

1. Confirm which copy of the site should use the license and review the activations in the AffStackDev account.
2. If the problem is an accidental change to the original site's security configuration, restore the correct configuration where appropriate. Otherwise continue with the explicit recovery action.
3. Tick **Clear this copy's local license credential and reconnect**.
4. Select **Prepare reconnection**.
5. When the screen says it is ready, enter the Dynamic Pricing key and activate again.

A moved or cloned site receives a new installation identity after confirmation. Credential recovery on an otherwise matching site keeps its installation identity. Both forms of reconnection clear this copy's saved license credentials and accepted local state so it can activate again. They preserve pricing rules, settings and order evidence.

Reconnection does **not** release the original site's remote activation and does not contact it. Manage that original activation in the AffStackDev account to avoid leaving an unused site slot occupied. A healthy matching installation does not offer this as a routine reset; use Check now or Deactivate license instead. If only the cached license state is damaged and the credential remains readable, use Check now rather than reconnection.

If recovery reports that local storage could not be saved, correct WordPress database/option storage first and retry. If encryption support has disappeared, restore the host's supported encryption method or prepare reconnection as appropriate. Do not expect a copied encrypted credential to work after its site binding or security salts have changed.

Release this site's activation

Use the license screen's remote-release action when you want to free this site's activation. Deactivating or deleting the WordPress plugin is different: those actions stop its scheduled license checks but leave the remote activation and retained records in place.

1. Open the license page on the correctly bound site.
2. Resolve any pending activation through **Check now / recover activation** first.
3. In **Release this site's license activation**, tick **Release this site's remote activation**.
4. Select **Deactivate license**.
5. Confirm that the current status becomes **Site deactivated**.

The release section appears only when the active credential is readable and the site identity matches. If the request is interrupted, the credential and original release request are retained; **Retry pending deactivation**

retries that same intent. Automatic recovery can also retry it. Permission may remain usable within its existing limits while confirmation is pending, but the remote slot must not be described as released until confirmation arrives.

A moved site cannot release the old binding through this control. Prepare reconnection for the current copy and manage the original activation through the account. Confirmed release removes the local active/pending credentials and scheduled validation, retains the installation identity, and keeps all saved business records. You can reactivate later with a valid key; there is no undo that silently reclaims the remote slot.

Signed plugin updates

Dynamic Pricing uses its own protected update service and verifies the publisher's signature before accepting a release. Its current license must be accepted and the service must have enabled installation-token updates for that installation. Local verification also needs the bundled public key, PHP Sodium signature support and ZipArchive. The private signing key never belongs on your store.

Signed plugin updates label	Meaning and next step
Activate your license	Recover current Dynamic Pricing permission and its active credential first.
Waiting for update service	The license is accepted but the service has not yet enabled this installation's protected updates. Use Check now after service availability is resolved.
Update verification needs attention	Ask the host to enable missing verification extensions and restore the authentic plugin package/public trust key if necessary.
Signed updates ready	WordPress can check for signed releases. This does not automatically opt the plugin into installation.

An administrator who can also update plugins sees **Check for plugin updates**. It refreshes the protected update check and, on success, WordPress's plugin update information. **Last update check** shows its UTC time and **Up to date**, **Update available** or **Update check needs attention**. This timestamp is separate from the license-check timestamps and is not an update-installation history.

Use **WordPress → Plugins** to enable or disable automatic updates. Manual and automatic updates use the same protected download and verification path; Dynamic Pricing does not force auto-updates. It offers stable releases newer than the installed version, and checks the release's PHP, WordPress and WooCommerce requirements before downloading/installing it.

Routine update metadata is cached for 15 minutes, including a cooldown after failures. The explicit update-check button requests a fresh check. The download and final installation also require fresh authorization, so an old update notice does not guarantee that an installation remains authorized. If the offer, license, site identity, trusted key or selected artifact changes, the update stops and must be checked again.

Recover an update that stops

First read the license status. A protected update request can also receive an authoritative denial of the installation license; correct the account or activation before retrying. A download-token failure by itself does not prove that the installation license was revoked.

Problem	Recovery
Service unavailable or throttled	Restore connectivity or allow the throttle to settle, then use Check for plugin updates. Routine checks have a short cooldown.
Host below the release's requirements	Meet the stated PHP, WordPress or WooCommerce minimum before retrying.
Temporary ZIP cannot be created	Have the host correct temporary-file storage and filesystem permissions.
Signature, archive or extracted-file verification fails	Obtain the intact publisher-signed package. Do not modify the proof files or force installation through a failed check.
Installed signed release has been modified	Review custom changes and unexpected files in the plugin directory. Reconcile them with the authentic signed installation before using the protected updater.
Offer or authorization changed	Refresh the license and update check, then retry the current offer.
Site has moved or credential cannot be read	Complete explicit reconnection and activation first.

For signed installations, the updater checks the entire installed file inventory, including unexpected files and symbolic links, so it does not silently overwrite a modified signed release. An existing installation with neither signed proof file can receive its first signed update; an incomplete proof pair is not treated as an unsigned installation. The downloaded ZIP, its embedded proofs and every signed file must agree, and the final unpacked source is checked again before replacement.

The protected service uses temporary download authorization; an update's displayed package selector is not a reusable download credential. Interrupted downloads are cleaned up locally. If a server or package problem persists, retain the exact error and consult the technical license/update error reference when requesting support.

What license data is retained

The store keeps its installation identity, bound website and environments, encrypted active/pending installation credentials, accepted plan/features/deadlines, bounded retry facts and the latest update-check result. It does not store the complete activation key or an AffStackDev account password. The account link opens the account site; it does not create a second account session inside Dynamic Pricing.

License operations send the bound site details and installation identity to the AffStackDev licensing service. Activation also sends the full key once and the proposed installation credential. Later validation, remote release and protected update requests use the installation credential. Downloading an authorized release uses a short grant credential. These credentials are not displayed on the license screen or included in its configuration export controls.

Changing the site's security salts or binding can make encrypted credentials unreadable. Use the supported recovery path instead of editing encrypted database values. A local storage error means the requested state could not be confirmed; resolve the storage problem before assuming activation, deactivation or reconnection completed.

WordPress deactivation and uninstall clear the scheduled license event but preserve local licensing data, rules, settings and order evidence. They do not release the remote site slot. A confirmed **Deactivate license** releases

that slot and removes local credentials while retaining identity and business records. **Prepare reconnection** clears this copy's local licensing state and credentials, and replaces the identity only for a moved/mismatched installation; it does not release the remote slot. These distinctions let you choose the appropriate operation without losing saved pricing work.

For storage fields, supported integration hooks and exact protocol limits, see the technical storage reference.

Dynamic Pricing's Core connection status

Open Dynamic Pricing [Settings & integrations](#) and find [Integration status](#). **Core runtime host** shows the selected runtime's status, host and version. **Commerce Core** separately shows whether Dynamic Pricing can use the required customer-eligibility and cart-line interfaces. A loaded runtime can still have a missing or incompatible business provider; the host row does not approve a customer or price change.

The usual host result is `loaded`, followed by the product supplying Core. `preloaded` means an existing runtime was already present. `same_version_conflict` means copies claiming the same selected version contain conflicting runtime files; restore matching supported product packages. `platform_unavailable` means the server or WordPress/WooCommerce requirements are not met. `no_compatible_copy` or `unavailable` requires restoration of a valid bundled or installed runtime. `not_ready` means selection has not yet run in that request. `site_mismatch` means the loaded runtime belongs to another WordPress site and cannot be reused for this site's operations.

When Dynamic Pricing is the selected host, it initializes the shared records automatically. Its own failed setup is deferred for five minutes before another request can retry; there is no merchant setup or reset button. Core also has its own shared-storage initialization. Preserve existing records when repairing a database problem. If another product supplies the runtime, that selected runtime and host handle their own setup.

Dynamic Pricing does not automatically take over another product's saved rules, gifts, customer-referral rewards or unfinished effects. It registers no general shared-feature executor. The bundled Core can coordinate compatible adapters supplied by other products, but those features require their complete supported contracts and current permission. There is no runtime selector, general feature-transfer control or Core JSON download button in Dynamic Pricing's Settings & integrations. The host-provided panel and local integration capabilities below are available only when the corresponding host or adapter exposes them.

What Commerce Core does

Commerce Core connects compatible products to shared customer context, cart-line identities and accepted order records. Dynamic Pricing includes the runtime and offers it for selection when its requirements are met. It needs 64-bit PHP 8.1 or newer, WordPress 6.7 or newer, and active WooCommerce 10.0 or newer. The included Core release is 0.1.0.2c; its technical version is 0.1.0.2.3. See [Dynamic Pricing's Core connection status](#) for the product's own Integration status display.

When several compatible products include Core, the highest eligible runtime version supplies the code for that request. Matching versions must contain matching listed runtime files; conflicting copies are held until the affected products are updated to a consistent release. An already loaded standalone Core is not loaded a second time. The **Core runtime host** row in Integration status identifies the selected runtime status, host and version. The selected host is not automatically the owner of your prices, gifts, business policies or credit balances.

Core coordinates information and records. Your connected products still control their business features and licensing. Installing Core alone does not create a discount, assign a customer to a business program, issue credit,

transfer a gift rule or add a new payment action. Features that need a product adapter stay unavailable until that compatible adapter is present.

Read shared records and technical details

Dynamic Pricing's Settings & integrations screen displays the host and consumer connection separately, as described [above](#). The bundled Core also contains a detailed panel that a compatible host can expose. When AffStackDev Suite supplies that interface, open **AffStackDev → Suite Settings → Commerce Core**, then expand **Shared records and technical details**. Viewing that host panel and downloading its records requires permission to manage WooCommerce and access to the host. The forms work without JavaScript. Dynamic Pricing does not independently add this panel or register its download action.

Display	How to read it
Commerce Core version	The version of the runtime selected for this request. A compatible connected product may supply a newer runtime than Dynamic Pricing's included copy.
Foundation — Available	Core can read its local runtime configuration. Connected products still have their own readiness checks.
Foundation — Needs review	Core cannot read the required local record. Resolve the database or stored-record problem before relying on affected shared features.
Audience connection — Configured	Core remembers at least one product responsible for determining consumer-promotion eligibility. This is a retained requirement, not proof that the product is currently working.
Audience connection — Not configured	No audience owner has been configured in Core's retained record. Each product continues to apply its own ordinary consumer behavior and other conditions.
Order records	Count of accepted Core order snapshots. If the database count fails, the card can display 0; the downloaded status report distinguishes that failure from an actual empty history.
Audience providers	Currently registered owner names, their declared versions and registration state. This list does not itself run their eligibility checks.
Shared feature owners	The last saved feature assignment, with its feature ID, capability, owner and current recorded state. Products verify permission and availability again before applying a change.

The shared-feature state reads **Operation needs reconciliation or is in progress** when an operation is pending, **Enabled** for an enabled unarchived definition, or **Paused** otherwise. **No assigned owner** means no owner is recorded; **A shared record needs review** means the returned entry cannot be presented normally. If no shared adapters are available, the panel explains that none are registered for this request. The list displays the first 25 shared records. An integration can obtain additional records through the paginated status interface; there is no Next button in this panel. The [shared-feature chapters](#) explain ownership, editing and operations.

Respond to connection messages

Connections need attention can show more than one issue. Resolve each applicable condition rather than treating the Foundation card as approval for every feature.

Message concerns	What to do
Local records unavailable	Restore database access and readable Core storage. Reload the panel after repair. Existing records remain the basis for recovery.
B2B needs its Commerce Core reader update	Use a B2B integration that supplies the supported fresh reader. The older reader can retain business exclusions but cannot approve uncertain consumer promotions.
Dynamic Pricing has multiple audience callbacks	Have the owning integrations remove the competing adapter as part of their coordinated update. One audience decision path is required.
More than one expected audience owner	Review the competing integrations and their retained requirements. Core does not choose an owner by loading order or silently discard one.
Previously configured audience provider missing	Restore that product's supported integration or follow the explicitly supported policy-retirement procedure for the affected consumer. Simply deactivating a provider does not erase Core's expectation.
Previously configured licensing protection unavailable	Restore the expected protection integration. A missing reader does not turn protected or unknown merchandise into ordinary merchandise.

A compatible Suite host can also provide its own product connection report. Accepted retirement is a separate proof that integrations may explicitly consume. This bundled Core's ordinary audience and line-protection readers do not automatically clear their remembered requirements when a retirement proof is accepted.

Download reports and individual order records

When a compatible host exposes the Core panel described above, open its **Records and support** section:

1. Select **Download status report** to download `commerce-core-report.json` with current runtime, provider, connection, snapshot-count and shared-feature information, plus the latest retained diagnostic sample.
2. To investigate a specific order, enter its positive numeric **Order ID** and select **Download order record**. The server accepts an existing order ID of 1–13 digits, without a sign, decimal point, spaces or leading zero.
3. Keep the file with the relevant support or integration investigation. Downloading changes neither the order nor its journal state.

Both downloads use the same filename. Rename saved copies if you need to retain more than one report. An individual order download contains the journal's revision, state, fingerprint and recorded data. It can include a draft or an interrupted publication as well as a committed record. It does not convert an unfinished record into an accepted one.

Enter a valid order ID means the input format or order lookup failed. Correct the ID and confirm the order exists. **This order has no readable Core record** means there is no readable Core journal snapshot for that order. A real WooCommerce order can exist without a Core snapshot, for example when no compatible historical-audience provider was configured. The export does not substitute today's customer classification or an older B2B record. **Access denied** requires a user with WooCommerce-management permission. If a form's security verification expires, reload the Settings screen and use its fresh form.

Reports are administrative records. Status reports identify integrations and saved shared definitions; order reports contain the recorded order subject and line evidence. Provide the relevant file to the person authorized to investigate the issue.

Customer eligibility and pricing integrations

An audience owner determines whether the current shopper is eligible for consumer promotions. Core ties its decision to the store, shopper and current checkout context. Changes in login, cart contents or quantities invalidate the request's previous context. Currency, shipping choice, tax scope, variation attributes and applied coupons also contribute to that context. Core passes a digest rather than storing the shopper's address in these context fields.

Fresh decisions have a limited validity period of at most five minutes and must match the same request. Consumer eligibility requires a consumer classification. Invalid, missing, conflicting or expired evidence leaves eligibility unresolved; it is not a permission to apply a consumer promotion. Connected products may also have their own finality and business-rule requirements.

The **Configured** state is deliberately retained when an audience provider disappears. Removing the plugin does not make all shoppers consumers. If no owner has ever been configured, Core reports that no audience provider applies; the connected product decides how its own standalone behavior uses that result. Core does not expose a customer selector, manual audience override or force-eligible button.

Legacy B2B and Dynamic Pricing

The older B2B reader can preserve a known ineligible outcome. It cannot establish fresh eligibility for a consumer, even when its earlier result said eligible. Its historical reader also cannot prove that an order's business context was atomically finalized. Install the supported reader integration when the affected product requires fresh or accepted historical evidence.

Core's Dynamic Pricing adapter runs only when an audience dependency already exists, B2B is present, or the runtime configuration cannot be verified. It requires exactly one audience callback. It delivers fresh supported owner evidence to Dynamic Pricing; it does not calculate the discount itself. Competing callbacks or uncertain B2B evidence hold the affected consumer decision until the integration is corrected. Installing Core by itself does not turn a previously independent Dynamic Pricing setup into a new audience-dependent setup.

Commercial facts without price changes

A compatible product can attach a commercial fact to a cart line, including currency, quantity, tax basis, price source, lock state, an optional unit amount and the kinds of adjustments its contract recognizes. Refreshing that fact replaces the prior fact in the same owner/slot. Core records the information; the owning product must actually apply any permitted price or adjustment.

Stored facts can remain on an order after their current-request validity expires. They describe what was recorded and do not approve a later price change. Missing amounts are distinct from zero. For integration authors, value rules and commercial-fact fields give the exact decimal, field and size requirements.

Shared cart lines and protected merchandise

Compatible products can opt a cart line into Core identity. An untouched ordinary line does not receive an identity merely because Core reads it. Once a product claims or links a line, Core preserves its UUID, the declaring product's reference and any declared relationships or facts. This allows connected products to describe a line consistently without changing the product itself.

A line can contain up to 16 provenance entries, 32 relationship entries and 16 commercial facts; Core cart operations support at most 200 lines. A product updating its own owner/slot replaces that entry rather than

adding another identity. A relationship creates matching outgoing and incoming records on two different lines. Removing that particular relationship keeps the lines' identities and other records. These are integration operations, with no general claim/link/unlink buttons in Dynamic Pricing Settings & integrations.

If Commerce Bridge supplies the existing line identity, Core reads it as Bridge-owned and does not migrate it. While the Bridge cart reader is present, Core's own line writers stay read-only, including for a line that Bridge describes as unprotected. Core also refuses an unknown protection result. An integration must establish one supported writer rather than let two systems alter the same identity graph.

Core reports relationships pointing to a cart line that is no longer present; it does not silently remove them. Complete order snapshot creation requires every related UUID to exist in the order. Duplicate IDs, mixed Core/Bridge identities, malformed records or unresolved protection need correction by the owning integration. If checkout displays **A shared cart line needs review. Please refresh the cart before checkout.**, refresh and review the cart. If the same message returns, investigate the line-owning and protection integrations before repeating checkout.

Core's classic-checkout line hook stores the validated line record in WooCommerce order-item metadata. Blocks snapshot capture reads whichever Core line records actually exist on its order items; the bundle does not register a separate Blocks line-record writer. Normal formatted order-item metadata and normal WooCommerce REST order responses omit raw `_afsd_cc_` records. Authorized Core readers and exports remain the way to inspect their recorded evidence.

Accepted order history and publication recovery

An accepted Core snapshot preserves the owning provider's committed audience decision together with the recorded order and line context. It includes the recorded customer, currency, totals, payment-method identifier and line evidence. The order fingerprint binds those fields to a specific order state. A snapshot does not itself mean the order was paid or authorize a commission, reward or credit.

Automatic capture is available on classic and Blocks checkout only when exactly one expected audience owner has a registered historical-order reader. A current eligibility reader alone is insufficient. If no such owner is available, Core does not manufacture a snapshot. This means the Order records count need not match your total WooCommerce order count.

Once accepted, history remains readable even if the customer changes, the order is refunded, the provider is removed or the order's current totals change. Those later events do not rewrite the original snapshot. Its purpose is retained historical context, not a fresh entitlement for today's shopper. An older B2B record can be returned only as unverified legacy history; it does not become a committed Core record.

Draft, publishing and committed

State	Meaning and next action
Draft	Core has prepared matching order and owner evidence but has not finalized publication. A supported integration can continue using the current revision and original order fingerprint.
Publishing	Core saved the durable publication intent before confirming the corresponding order metadata. Preserve that intent and investigate the original record.

State	Meaning and next action
Committed	The exact order projection was confirmed and the journal finalized. The accepted record remains available for historical reads.

If the panel reports **An interrupted order publication needs review**, download the original order record. A user with WooCommerce-management permission can have the supporting integration invoke Core's reconciliation API with that exact order, revision and fingerprint. There is no reconciliation button in this panel. Reconciliation confirms that the intended order projection already exists and then finalizes the journal; it does not write a second projection or replace the original request.

If that projection is missing or differs, the operation remains pending. Passage of time does not authorize a takeover or discard the intent. Resolve the saved-record discrepancy through the owning integration. Do not create another order or use a different operation identity as a substitute for confirming the original result.

At checkout, **The order context needs review before payment** means Core could not verify the order fingerprint. **The accepted order context could not be saved. Please contact the store before retrying payment** means preparation or publication failed. Inspect the existing order and its retained Core state before retrying payment. A late failure can leave a publishing record that requires reconciliation rather than a fresh attempt. Technical procedures are in snapshot publication and snapshot recovery.

Diagnostic reports and integration interfaces

The status report includes diagnostic counters from the current request and the latest retained affected request. Core retains up to 32 distinct bounded code names per request, with each counter capped at 999. It saves one latest sample rather than an accumulating activity log. An ordinary "no audience provider" outcome is not stored as a persistent warning. A later request with no new relevant codes leaves the earlier sample intact, so compare its recorded time with the current problem.

Diagnostic samples contain code names, counts, Core version and recording time rather than raw provider error text or shopper details. The broader status and order reports contain their respective integration and order records. A report code identifies the failed boundary; it does not mean every previous step was rolled back. Runtime error recovery and [shared-operation recovery](#) explain how to interpret the retained state.

Developers use Core's local PHP interfaces to register readers, read current audiences, manage line declarations, prepare and reconcile order snapshots, and call the separate shared-definition, operation, retirement and checkout-participant protocols. There is no public HTTP audience endpoint. Most local interfaces rely on the calling integration to verify its own user permissions and request security. Core adds WooCommerce-management checks to its administrative downloads and specified management/reconciliation operations. The complete function reference identifies every facade, argument order and owning contract.

The selected runtime supplies the `affstackdev-commerce-core` translation domain. The included catalog is a template that translation packages can use. Core's status layout wraps long identifiers, stacks cards and download fields on smaller screens, and provides visible keyboard focus for its controls. Its included GPL text is available with the runtime files.

Retained commerce records and safe updates

Commerce Core keeps the records that let compatible products agree on a feature's current configuration, responsible product and unfinished operation. It does not become the owner of product prices, customer balances or payout ledgers. Orders continue to use WooCommerce's normal storage interfaces.

Dynamic Pricing includes Core 0.1.0.2c. Another compatible active host can provide the selected shared runtime. The available operation depends on that selected runtime and the owning products' supported interfaces, rather than on a release number alone.

Records belong to the original WordPress site. Core requires a supported 64-bit PHP environment and rejects attempts to reuse its active runtime after switching to another site. Its installer creates missing local records without rewriting an existing database schema. Network-wide installation through the Core installer is refused; activation must be handled separately for each site. An unavailable or unfamiliar record is preserved and held for recovery.

Each record has an identity, a revision, a state, its data and a fingerprint. A successful update must name the revision it actually read. If someone else saves first, the stale update fails instead of overwriting their work. A new record uses expected revision zero. For example, if two operators read revision 7 and one saves revision 8, the second must reload revision 8 before making a reviewed change. Saving at revision 7 again does not overwrite revision 8.

A fingerprint confirms the expected data content; it is not a payment receipt. Core can coordinate its records without guaranteeing that a separate financial service completed a payment. If a result is uncertain, keep the original reference and reconcile the owning service before taking another value-changing action.

Core also remembers which audience and line-protection services it has already depended on. Adding a requirement is persistent. Removing the product does not erase that requirement or make a previously protected benefit safe. There can be at most 16 expected audience owners. Repeated conflicting configuration writes eventually return a busy result; they do not silently drop an owner. Snapshot status counts report retained records by state, not the amount of money owed or paid.

These records have bounded, validated data. The journal accepts at most 128 KiB of canonical data per record; individual text values are at most 2,048 bytes, lists/maps at most 400 entries per level and nesting at most 12 levels. The responsible integration supplies supported fields and types. Merchants should use its current form or supported service instead of editing database rows or changing revision numbers manually. Exact storage contracts are in the technical journal reference.

Choosing a supported feature provider

A shared feature can run only through a product that explicitly supports its full configuration and is currently ready to perform the requested task. Products register their local adapters during normal initialization. This happens automatically through code supplied by those products; there is no screen for entering a PHP callback.

Provider registration identifies the owner, provider version, supported feature families and schema versions, preference priority, and separate availability, evaluation, execution and reconciliation functions. Up to 16 owners can register; an owner describes up to 16 feature families with up to eight schema versions each. These limits are integration boundaries, not a reason to divide a financial operation across several invented owners.

Core distinguishes checking or editing a feature, executing its effect and reconciling an earlier effect. A product may remain able to read an old receipt while it cannot perform new business. Both full feature support and

current task readiness must succeed. Missing callbacks, exceptions, a wrong site or conflicting registration keep the feature unavailable. A duplicate accepted owner can make the shared registry unavailable for the whole request; resolve the conflicting integration and begin a fresh request.

The ready preferred owner wins. If the preferred product is unavailable, Core considers other products that explicitly support the same definition. Among those, a smaller provider priority wins, with owner-name order breaking ties. For example, priority 10 precedes 200, but a ready preferred owner takes precedence over both. This is a technical adapter setting; it is not a ranking of merchant product discounts. A feature's own stored priority is separate from provider priority.

The provider overview reports registered or unavailable and its declared feature support. Registered does not mean licensed for this transaction, eligible to serve this customer or safe to issue money. Those checks remain with the product.

An authorized integration can retire a shared executor, or restore its availability, using the exact saved retirement-record revision and a reason. This requires WooCommerce management permission. Retirement prevents that owner from beginning a new authorized effect; it does not delete shared definitions, undo earlier effects or remove its ability to supply supported old-receipt reconciliation. A compatible alternative may become responsible after current authority is resolved. A pending operation is not forcibly transferred by this action. This executor control is separate from the [business-policy retirement procedure](#).

The technical provider contract defines the trusted local integration fields. It does not add a general provider editor to Dynamic Pricing.

Shared configuration, pauses and ownership

Shared configuration keeps one original feature identity while compatible products can edit or perform that feature under its supported contract. A product must understand the complete saved terms. Installing another product does not convert old terms or turn on a paused feature.

Configuration or record	Effect and restriction
Feature identity	The original unique ID is retained across edits, ownership changes and recovery.
Capability and semantics	Identify the feature family and exact meaning of its terms; existing records cannot change these identities.
Definition schema and payload	The owning product's complete supported field set. A replacement schema must be understood explicitly. Core does not infer missing settings.
Enabled	Allows execution when true and every other check succeeds. Turning it off retains the definition.
Archived	Holds execution while preserving configuration and history. It does not cancel a customer's accepted obligation.
Feature priority	Integer 0-9999 retained for the feature's own meaning; not the provider selection priority.
Preferred owner	Empty for normal supported selection, or the exact preferred supported product.
Creator and last editor	Preserve the original creator and identify the most recent changed definition.

Configuration or record	Effect and restriction
Definition revision	Advances when the complete configuration changes.
Ownership generation	Advances when configuration or selected ownership changes.
Record revision	Used to protect every saved replacement from a stale request. It is separate from definition revision and generation.
Pending operation	Holds the original operation reference, attempt token, expiry and evidence; blocks another feature operation or edit until resolved.

These fields are supplied through an owning product's form or an authorized local integration. Dynamic Pricing does not provide an unrestricted editor for arbitrary feature payloads. Configuration data must fit 48 KiB and the common text/depth limits.

An editor must currently support editing that exact definition. Saving a changed definition requires its exact record revision and no pending operation. It preserves the creator, advances definition revision and generation, clears the old owner and records the last editor. A save with identical terms leaves the row unchanged, but still checks edit authority and current revision. There is no shared-feature delete operation in this interface; disabling or archiving preserves the record.

When a compatible product asks who should act, Core checks whether the feature is enabled, unarchived and free of an unfinished operation, then selects a ready supported owner. If none is available, the saved feature remains held, potentially with an empty current owner. A missing owner never authorizes another product to guess the terms. Simultaneous ownership changes may return busy after three conflicting attempts; reload the current result.

The shared status view is read-only and requires WooCommerce management permission. It shows owner, generation, definition revision, enabled/archived state and whether work is pending. An unreadable feature is identified as unreadable rather than hidden as successful. Reporting is paginated: 25 records by default and at most 100 per request. A displayed owner is a stored assignment, not permission to execute now. For exact output fields see shared definitions and reporting.

Fresh evaluation and short-lived permission

A product evaluates a shared feature using its current configuration, current owner and exact transaction context. Core does not retain an old cart result as permission to apply it later. The selected product supplies the result, and Core checks authority again after that callback completes.

The returned authority claim lasts at most 60 seconds. It binds the site, feature, owner, ownership generation, definition revision and complete definition/context fingerprints. A changed customer or transaction context, changed owner, edited definition or expired claim invalidates the earlier evaluation. Obtain a fresh evaluation instead of editing or extending the old claim.

An unfinished operation has its own reserved authority and prevents ordinary resolution from supplying a competing owner. A local claim is not a public authentication token, product license, customer eligibility decision or financial receipt. The relevant product must still enforce each of those requirements.

Evaluation results are bounded to 16 KiB; context is at most 8 KiB. The owning product controls the supported fields. Integrators can use the evaluation and claim reference to preserve the complete envelope and correct data types.

One business operation, one retained identity

Every delegated operation uses a stable business reference and exact context. Core derives a stable operation and effect identity from that original reference and site. Keep it when retrying, moving between supported owners or investigating a failure. Creating a new reference to get around a pending result can create a separate effect.

Operation state	Meaning and next step
Pending	An attempt was reserved and its outcome is not yet confirmed. Reconcile the original operation.
Committed	The owner supplied a confirmed effect receipt. Repeating the original request returns the retained result without executing again.
Not applied	The owner supplied a confirmed no-effect receipt. Ordinary repeats return that result. Another attempt requires the separate explicit recovery conditions.
Unknown callback answer	The journal remains pending; unknown is not a fourth terminal state or proof that nothing happened.

Core reserves the feature before asking a product to perform its effect. The feature permits one pending operation at a time. The operation freezes the feature definition, context, owner and short-lived claim, with a unique attempt token and empty receipt. Other operations and edits remain held while the reservation is unfinished.

Immediately at its financial action boundary, the product must confirm that the original pending record, token, lease, owner, generation and definition revision still match. Core checks this before and again after callback-based readiness. Retiring the owner or losing its current availability can stop a stale attempt. The product still owns its transaction, money lock, durable deduplication and authoritative receipt; a Core check cannot replace those controls.

A product returns a confirmed applied or no-effect result with its receipt. Core saves that terminal result before releasing the feature reservation. If reservation release fails, replaying the terminal original result can retry that release safely. A thrown callback, malformed answer or unknown outcome preserves pending evidence. A storage error can happen after the owning product applied an effect, so do not equate an error with “nothing was paid.”

Passive lookup retrieves the original record without executing anything. It is useful for support and reconciliation, but its existence alone does not make a pending effect safe to repeat. The technical operation contract defines the exact callback and receipt fields.

Reconciling and recovering unfinished work

Reconciliation asks a supported authoritative reader what happened to the original operation. It never invokes the effect callback again. Core waits until the original reservation's lease expires before reconciling pending work. If the process stopped after reserving the feature but before inserting its operation row, Core can reconstruct that original pending row from the reservation, keeping the same claim and token.

The reconciler may be another supported product that understands the frozen feature and can verify its authoritative original receipt. It must report confirmed applied, confirmed not applied, or unknown. Unknown

keeps the hold. Expiry by itself does not prove absence: an earlier writer could still have applied the effect before the response was lost.

A new attempt is supported only through the explicit no-effect recovery API, when both the selected Core and the owning integration support it. The embedded Core does not create an automatic financial retry schedule or unrestricted Retry button for this API. The operator uses the original product's supported recovery workflow.

For explicit recovery, all of these conditions matter:

1. The original operation is confirmed **not applied**, its old claim has expired, and the request names its exact current operation revision, receipt and unchanged context.
2. The current feature's full definition and definition revision match the frozen original. Editing terms and then restoring their old values still changes the revision and blocks this recovery route.
3. An authoritative reconciler freshly confirms the same no-effect receipt and proves that every old writer is prevented from applying the old attempt. An empty table, elapsed timeout or manually typed receipt is insufficient.
4. A currently ready supported owner receives a different attempt token and bounded authority. The original business reference, effect identity, site and terms remain.
5. The old terminal attempt is archived before the original operation is advanced to pending again. Only the winning current revision can reach the effect callback, which must still enforce its own financial protections.

For example, after a provider confirms that reference reward:order-42 was not applied, recovery can keep that same reference and its original frozen terms. Submitting reward:order-42-retry would be a different operation and is not the supported recovery procedure.

If recovery is interrupted before advancing the operation, the same request can resume an existing matching reservation without extending its expiry. An expired reserved attempt cannot be executed merely by resuming it; it must be reconciled. A failure after the new pending operation is saved leaves pending evidence. Reread that operation after an ambiguous result or conflict rather than repeatedly pressing retry or clearing another attempt's reservation.

Archived attempts retain their exact original receipts and payloads, linked by previous-attempt revisions. They are not overwritten or embedded recursively. Keep the archives, current row and financial owner's evidence together. If the current host lacks the required recovery interface or the original terms changed, leave the original case held for supported owner review; there is no force override. Exact requirements are in same-operation recovery.

Retiring a business policy safely

Business-policy retirement is a specific proof that an owner has no live policy left in an exact scope. It differs from retiring a shared executor. The two supported scopes concern current audience policy and protected-line policy. A compatible Suite host can expose B2B audience retirement controls. Dynamic Pricing shows the accepted removal proof and can use it only with its explicit, separately checked consumer-only pricing policy; it has no retirement form. Host controls do not expose every owner/scope or release the B2B product's business freeze.

Use the owning product's preparation procedure before deactivating it. The product must establish a durable freeze and confirm no live policy remains. Core records the exact owner, scope, site, request identity, policy revision, evidence reference and freeze reference. Preparation also records the current runtime configuration

fingerprint and a five-minute acceptance deadline. Core checks the proof's shape; the owner is responsible for the actual freeze.

Step	Requirement and effect
Prepare	Owner and its retirement guard are present. An authorized WooCommerce manager uses the current revision. The guard confirms no live policy and frozen state. Core writes Prepared evidence.
Deactivate owner	Follow the owner's supported procedure while keeping its freeze intact. Preparation alone does not remove the product.
Accept	Use a fresh request after deactivation, before five minutes expire. The guard and both site/network-active plugin entry must be absent, with the exact Prepared revision and unchanged runtime fingerprint.
Read accepted proof	A compatible consumer freshly verifies the accepted original record. It does not substitute the management display or an error.
Cancel	An authorized exact-revision write revokes the record. The owner separately decides when its own resume conditions are satisfied.

An attestation can freeze owner-side work before Core's final save succeeds. If preparation fails, keep the freeze and check the actual stored result; do not assume the owner can resume. Likewise, an acceptance notification can fail after the Accepted record was saved. Reread persisted state instead of repeating a different operation on the assumption of failure.

Accepted proof does not expire simply because the original five-minute preparation deadline passes. It remains subject to its site, runtime fingerprint, absent owner and unrevoked state. Changed runtime expectations, returning owner or invalid data make it unusable. Retirement does not erase old audience/protection requirements, transfer business prices, change historical order evidence or grant consumer discounts. Ordinary Core readers still hold when their required provider is absent unless a compatible consumer explicitly uses the separate retirement proof under its own rules.

The management view separates **stored state** from **validity**. State can be None, Prepared, Accepted or Revoked. Validity can be none, current, expired, stale, owner present, revoked or unavailable. Prepared can be current while the owner is present; that does not make acceptance possible yet. The owner-present flag reflects the collected guard or retained plugin basename, not an unrestricted scan of every possible owner.

Revocation reporting also distinguishes not attempted, confirmed and unconfirmed. This is what the current request observed; a later stored revision may differ. A read/write error must not be shown as confirmed revocation.

Cancellation always writes and advances the revision, even if the record was already revoked. Expired or stale proof can be cancelled, but corrupt or wrong-site records remain held for recovery. If no record exists, cancelling at revision zero creates a cancellation-only marker. That marker is not retirement proof; it prevents a delayed earlier preparation from appearing after cancellation. A later legitimate preparation requires the newer revision and new authentic owner evidence.

When an owner returns, Core attempts to revoke its previously accepted audience and protection records. It preserves Prepared records so the normal prepare → deactivate → accept sequence can work. Failed revocation remains unconfirmed: the owner's durable freeze must stay until all relevant cancellations succeed under its own write lock. A saved cancellation does not itself release that lock or resume business policy. Do not delete

the journal or reset a revision to bypass a failed cancellation. The technical retirement contract contains exact fields and permission boundaries.

Discovering checkout credit participation

Core can advertise the currently registered Store Credit reservation participant to a compatible checkout coordinator. This discovery is independent of ordinary audience providers and shared feature owners. It does not reserve or spend credit.

The supported descriptor has exactly six values: schema 1, owner `affstackdev-store-credit`, capability `store_credit_reservation_participant_v1`, integer revision 1, and the owner's callable `enlist` and `order-verification` functions. Compatible plugins supply this descriptor in code during checkout-participant collection at initialization priority 40. There is no customer field for choosing a callback or entering an alternate protocol revision.

Registration must finish before another plugin resolves the participant. A duplicate attempt for the supported owner makes the participant conflicted for the request, even if the second submission is identical or follows a malformed first attempt. A thrown collection callback discards partial registration. Records are not saved across requests, and stale unavailable callbacks are discarded rather than reused.

Connection status	Meaning and response
Registered	Descriptor and callable functions are available. The actual transaction still needs its owning products' checks.
Not ready	Collection has not finished; the integration must resolve later in initialization.
Missing	No accepted supported descriptor was registered; restore the owning integration.
Unavailable	Required function is not callable; restore the compatible implementation.
Conflict	Competing supported-owner registrations occurred; correct them before a fresh request.
Failed	Collection could not complete; investigate the registering service.
Site mismatch	The active Core belongs to another site; use the original supported site context.

Reading status does not invoke either financial function. Resolving a participant checks the exact owner, capability and integer revision, then returns a detached descriptor for the trusted caller. It does not validate an order's financial proof, verify payment, mark an order paid or provide a substitute participant. B2B owns coordination, Store Credit owns its ledger and receipt evidence, and both products' actual transaction contracts remain necessary. Dynamic Pricing does not register or call this participant. Its [coupon and credit combinations](#) decide whether pricing may coexist with recognized credit; that is separate from reserving or spending the credit. The technical participant contract describes this discovery interface.

Shared operation troubleshooting

Symptom	Appropriate response
Record or runtime cannot be verified	Restore the original site's compatible runtime/storage. Preserve unfamiliar schema and corrupt evidence for recovery; do not overwrite it as empty.
Save conflicts or reports busy	Reload the current revision and review concurrent changes before saving.
No supported feature owner	Check the exact definition, product availability, enabled/archive state, adapter registration and current permissions. A displayed old owner is not a grant.
Evaluation or claim expired	Obtain a fresh current evaluation; do not edit the token or extend its expiry.
Operation pending or outcome unknown	Keep the original reference and context. Reconcile through its authoritative owner after the lease boundary; do not submit a new financial operation.
Confirmed not applied but retry held	Review exact operation revision/receipt, unchanged definition and fresh no-effect proof. Explicit supported recovery is required.
Archived attempt cannot be read	Keep the original record/history intact and restore the compatible reader; do not manufacture replacement history.
Retirement preparation fails	Keep the owner's freeze, inspect current proof/revision and complete the owning product's supported preparation.
Acceptance is unavailable	Confirm owner deactivation, five-minute preparation window, unchanged runtime and exact revision. Reread actual state after ambiguous results.
Cancellation or automatic revocation is unconfirmed	Keep the freeze and inspect persisted state under the owner lock. Cancel at the current revision; do not assume it succeeded.
Retirement was accepted but benefits remain held	Acceptance is separate evidence, not a customer eligibility grant or an automatic clearing of provider requirements. Follow the consuming product's supported policy.
Checkout participant is missing/conflicted	Restore the exact compatible owner and correct registration timing/duplicates. Discovery failure does not authorize bypassing credit checks.

Provide support with the original site, feature/business reference and current error code. Keep the original journal contents intact while the owning integration resolves the case. The technical error lookup maps every returned code to its specific recovery boundary.

Use the complete installed references

The plugin installs two complete references for its installed version. The Customer Reference explains setup and operation. The Master Technical Reference explains exact fields, contracts, stored evidence, integrations and internal operating rules. Word and PDF editions of the customer reference are on affstackdev.com.

Download a reference

1. Open Dynamic Pricing → Settings & integrations.

- 2. In Documentation, choose Download customer reference Markdown or Download technical reference Markdown.
- 3. Save the attachment. A text editor, Markdown reader or model document reader can open it.

The same buttons appear in the retained-record screen when licensing pauses new pricing work. Your WordPress account must have `manage_woocommerce`, the capability used to manage this product. Downloading a manual does not require a pricing entitlement, calculate a cart or change configuration.

If WordPress rejects an old download link, reload the administration screen and click its current button. An unavailable-document response means the requested ID or installed file could not be accepted; use the provided button and restore the full current package if files are missing. A permissions response requires an account with the management capability. A read failure requires restoring readability of the installed manual.

Use the references with AffStackDev AI

The installed package identifies both manuals to a compatible AffStackDev AI inventory reader, including which is technical, which is for customers and which release each describes. AffStackDev AI 0.1.0.11a understands these explicit audience and edition fields. The reader can obtain the full canonical manuals rather than relying only on the short product description.

After updating Dynamic Pricing, refresh its entry in the inventory so the reader sees the current installed references. Refreshing reads files already installed; it does not install the update, enable a rule or authorize the model to change pricing. The product's separate merchant assistant controls still require their own permissions and explicit review steps.

Glossary

Term	Meaning
Rule	A saved pricing definition with its identity, revision, priority and enable state.
Offer	A rule's evaluated price plan for participating purchased units.
Bulk	One qualifying quantity-break rate applied to every matching unit.
Graduated	Rates beginning at specified numbered units in a stable allocation order.
Set price	One price for each complete group of N matching units.
Catalog price	The current product price used before the Dynamic Pricing offer.
Floor	Minimum unit price enforced before subsequent coupon deductions.
Cap	A per-rule, per-cart limit on units, applications or savings.
Audience	Authoritative customer classification and eligibility for consumer pricing.
Owner	The component authorized to decide a particular commerce effect.
Revision	Version of one saved record, used to detect conflicting edits.
Archive	Retained rule state excluded from normal execution until supported restoration.
Snapshot	Decision facts retained at one acceptance point for comparison or explanation.
Evidence	Saved pricing decisions and observed order financial information.
Retirement	Explicit release of a defined provider's authority under its supported contract.

Term	Meaning
Core	Bundled shared commerce coordination runtime with its own version and requirements.
Enhanced interface	Optional JavaScript administration layered on the PHP screens.
Nonce	An action-bound WordPress request token checked alongside user capabilities.

Use connected AI actions

Authorized administrators can ask connected AffStackDev AI to inspect Dynamic Pricing and prepare workspace changes. Dynamic Pricing supplies reviewed actions using the WordPress account that authorized the connection. **Me** refers to that account.

The assistant can create or edit a pricing rule, use any of the four templates, duplicate a rule, enable or pause rules, archive or restore rules, change priorities and restore a retained revision. It can change the engine, overlap strategy, audience policy, enhanced navigation, pricing labels, permitted coupon combinations, Store Credit combination setting and every shopper-display setting. Rule proposals support the native targeting, exclusions, roles, quantities, spend conditions, tiers, schedules, floors, caps and nested conditions described in this reference.

It can also import rules, save a current merchant-assistant draft, inspect order pricing evidence and reports, export saved configuration, read the installed manuals and run sample simulations. License checks, deactivation and preparation for reconnection use the native licensing client. Entering a complete license key and activating it remain a human step in **Settings → Dynamic Pricing license**; the assistant supplies that screen's link and can inspect the resulting status.

Review and apply a pricing request

1. Describe the desired result and name the products or rules. Product IDs and SKUs also work. For account simulations, use a username, email, name, **me** or **guest**. If several choices match, inspect the returned candidates and select the intended one.
2. Ask the assistant to prepare the exact change. Inspection and preparation do not save pricing rules, enable discounts, change settings, reserve an import or contact the licensing service.
3. Open the WordPress project review link. Check the resolved targets, current values, proposed values and effect descriptions. Creating or editing an enabled rule can change future cart pricing; a separate engine setting also controls whether live pricing runs.
4. Approve the exact proposal in WordPress. Approval and implementation are separate actions. Apply the approved project when its displayed effects are intended.
5. Read its result. Dynamic Pricing verifies the stored rule identity, revision, full definition and status, or the saved settings revision. A configuration receipt confirms those changes. Use a sample simulation or the owning checkout workflow to examine the resulting customer calculation.

A new rule needs a name; omitted fields use the native new-rule defaults, including a 10% offer on all eligible products, sale products excluded, and priority 100. It starts paused unless the proposal explicitly requests it enabled. Duplicates and templates start paused. An omitted duplicate name uses the source name followed by "(copy)"; restore an archive before duplicating it. Imported rules and restored revisions remain paused. Enabling

rules does not bypass licensing, customer eligibility, the engine switch, schedules, protected lines or coupon policy.

Rule edits change only the fields you supply and preserve every omitted value. Read the current rule before proposing an edit. Empty arrays explicitly clear selections, zero removes an optional date or cap, and `conditions_json` containing `null` clears additional conditions. Required native values cannot be cleared. Changing the target scope or attribute taxonomy requires explicit targets. Settings changes also preserve omitted values. Review the final values before approving either operation.

Find and inspect information

The action inventory identifies the installed Dynamic Pricing provider and its current input choices without requiring a prior administration visit or saved action cache. Initial license identity setup remains a native human step. Rule, product, customer, term and coupon searches return bounded pages with a next-page position. Rule searches support Working, Enabled, Draft, Archived, Retained reward and All status filters; an omitted AI filter returns all records. Integration inspection reads the native current-request status and compatibility requirements without authorizing checkout. A rule can be read with its current revision and resolved product or term references. Revision history has its own paginated view. Templates show their native initial definitions.

Order inspection reads current report figures, line pricing and financial metadata, checkout decisions and retained observation events. An event's decision fingerprint can select its historical decision. Large decision, report, configuration-export and manual contents are returned in numbered parts. Keep the parts in order and ensure their supplied checksum stays the same across reads before assembling a file; a changed checksum means the source changed during retrieval.

The simulator accepts the same scenario and account modes as the native screen. It uses actual catalog prices and the saved rules. A scenario can specify consumer, B2B or unknown eligibility and sample roles. Account mode resolves an actual site account and uses its supported current eligibility and retained history. Inspect the summary and then the paginated lines, rule traces or competing-offer explanations. Simulation does not change a cart, create an order or persist provider-detection flags.

Import and bulk operations

Bulk enable, pause, archive and restore can include up to 50 distinct rules. A bulk priority request can save up to 25 distinct rules with an individually entered priority for each, matching the native list. Single-rule priority and retained-revision restoration remain available. The review binds every selected rule and its current revision.

Direct import uses the native **skip**, **update** or **copy** conflict policy. Supply a Dynamic Pricing export containing the intended rules. The connected action envelope is smaller than the native file-upload form, so split large imports into smaller reviewed batches. If the complete review would exceed its bounds, the proposal is rejected before any rule changes.

For a native import preview, inspect its rows, then prepare each row by preview token and zero-based row number. Separate reviewed projects support all 100 native rows. Tokens belong to the current administrator and expire after 20 minutes. Changed, expired or completed rows require inspection and fresh proposals. Shared operation identities prevent duplicate saves if the native form is submitted afterward.

A current single-draft merchant-assistant preview can likewise be saved through a separately approved project. This adds an approved save path; the earlier assistant's preview request still cannot save or enable a rule by itself.

Bulk and import work saves one record at a time. A conflict or interruption can leave an incomplete batch. Inspect the individual saved records and reconcile the original project; the integration never automatically repeats the unfinished mutation to discover what happened.

Changes that require a fresh review

Changes to target revisions, source rules, pricing settings, selected products or terms, relevant roles or coupons, currency, tax context, site identity, active plugins, provider version or action schema require a fresh proposal. Capabilities and licensing are rechecked before saving. Earlier approval cannot authorize changed terms.

Repeated application of a completed project returns its existing receipt. If execution is interrupted, the original project remains uncertain until reconciliation reads the exact saved result. Reconciliation does not send a licensing request, recreate a rule or save another revision. An incomplete batch or a mismatched saved result remains unresolved. A new write may be held while that original operation is running or uncertain.

Once an interrupted action has a verified applied receipt, a separately reviewed next operation can retire its old write reservation. The integration never releases an uncertain reservation merely because time passed. If saved evidence cannot establish completion, use the native records and developer support to resolve the original operation.

Removing or excluding the plugin removes its executable actions. Historical projects and receipts remain available to their authorized owner. Reinstalling a different version does not make an earlier proposal valid; prepare it again against the installed version.

Licensing and companion effects

A license check contacts the trusted licensing service with the saved credential. It can recover unfinished activation or, as stated in its review, finish pending deactivation. Deactivation can release the remote activation and stop future pricing. Reconnection preparation clears this copy's local state when native recovery permits, without releasing the original site's remote slot.

A lost licensing response remains uncertain if completion evidence is unavailable. Observation-only reconciliation will not submit it again. The native licensing screen retains its explicit recovery controls. Complete keys, installation credentials and cryptographic material are never accepted as action inputs or returned to the assistant.

Dynamic Pricing owns purchased-item pricing; Free Gifts & BOGO owns gifts; AffiliPartnerCS owns attribution and commissions; Store Credit owns balances and issuance. Pricing receipts do not certify companion effects. The preceding ownership and eligibility rules apply in full.