TECHNICAL SPECIFICATION

Freshcart — GA4 dataLayer & Google Tag Manager

The complete measurement specification for the Freshcart storefront: every event, every parameter, the container configuration required to receive them, and how to verify the whole thing works.

GTM container	GA4 property	Currency	Events specified
GTM-5PDKGPN9	G-JSJ85T9LV1	GBP	21

1. Scope

Freshcart is a UK online grocery storefront. This document specifies its analytics implementation completely enough that a developer could rebuild it, and a tag manager could rebuild the container, without asking a follow-up question.

Every payload in this document is generated by running the shipped tracking code, not transcribed from it. The worked examples come from `src/lib/freshcart/specExamples.ts`, which calls the real functions in `src/lib/freshcart/datalayer.ts` against the real catalogue and captures what they push. A specification maintained separately from its implementation drifts, and it drifts silently in exactly the details that matter — an `index` off by one, a stale `item_category4`, a `discount` that no longer reconciles. Here that is not possible.

In scope: GA4 ecommerce, engagement and consent events; Consent Mode v2; the web GTM container; Google Ads conversion and remarketing. Out of scope: server-side tagging, and any non-Google destination.

2. Architecture and container isolation

Freshcart shares an origin with a second demo storefront (Roamio, a travel eSIM shop) that has its own GA4 property and its own GTM container. Two properties on one origin will cross-contaminate unless deliberately separated. Three rules keep them apart.

Rule	How it is enforced
One container per document	GTM and the consent defaults load from each site's own `layout.tsx`, never the root layout. `/freshcart` loads only `GTM-5PDKGPN9`; the hub page at `/` loads no analytics at all.
One dataLayer	A single global `window.dataLayer`, as Google documents. Freshcart does NOT use a custom-named dataLayer — the containers never coexist, so a shared array is correct and safe.
Namespaced storage	Every key Freshcart writes is prefixed `freshcart_`, including its own separate consent record. Consent given on one demo site must not silently apply to the other.

Navigation model

Freshcart is a multi-page application on purpose. Every in-site link is a plain `<a href>`. Every navigation is a real document load: the consent defaults re-run, GTM re-initialises, and the Google tag fires its own page view.

> **Consequence: no manual page_view**
> Because every navigation is a real page load, the Google tag's built-in page view (`send_page_view`, default `true`) is the only source needed and is already correct. **Freshcart pushes no `page_view`, and the container must contain no `page_view` event tag.** Either one alongside the automatic view double-counts every page.
>
> Roamio, next door, is a single-page app: its document never reloads, so route changes are invisible to the Google tag and it *must* push page views by hand. Two sites, two navigation models, two opposite correct answers. Copying either implementation to the other site breaks it.

The second consequence is that no in-memory state survives a click. Trolley, auth, slot and consent all live in `sessionStorage` / `localStorage` and are re-read on every page — which creates a hydration hazard that is covered in §14.

3. Event map

The canonical funnel, in the order GA4 expects to receive it. All 21 events are listed in Appendix A.

#	Event	Fires on
1	view_item_list	Aisle page, search results, home offers row
2	select_item	Product tile click
3	view_item	Product detail page
4	add_to_cart	Add, or stepper increase
5	remove_from_cart	Stepper decrease or removal
6	view_cart	Trolley page
7	begin_checkout	Trolley → slot commitment (see §6)
8	login / sign_up	The slot-booking gate
9	add_shipping_info	Slot confirmed
10	add_payment_info	Payment submitted
11	purchase	Confirmation page, once per order

4. The item schema

Every ecommerce event carries an `items[]` array built by one function, so the shape is identical everywhere. The four taxonomy levels are the reason a grocery catalogue is worth modelling: Category → Department → Aisle → Shelf maps exactly onto `item_category` … `item_category4`.

Parameter	Type	Required	Scope	Example	Notes
item_id	string	Required	Item	"fc-0015"	SKU. One of item_id or item_name is required; we always send both.
item_name	string	Required	Item	"Freshcart Chicken Tikka Masala"	As displayed.
item_brand	string	Optional	Item	"Freshcart"	Own-label lines are "Freshcart"; branded lines carry their brand.
item_category	string	Optional	Item	"Fresh Food"	Level 1 — category.
item_category2	string	Optional	Item	"Chilled"	Level 2 — department. A route segment.
item_category3	string	Optional	Item	"Chilled Ready Meals"	Level 3 — aisle. A route segment.
item_category4	string	Optional	Item	"Ready Meals"	Level 4 — shelf. Shared across products; not a unique key.
item_variant	string	Optional	Item	"400g"	Pack size. A documented, unremarkable use of the field.
price	number	Optional	Item	2	Unit price ACTUALLY CHARGED — the discounted price on a promoted line.
quantity	number	Optional	Item	3	Defaults to 1. On add/remove this is the DELTA (§5).
discount	number	Optional	Item	0.5	Per-unit saving. price + discount recovers the shelf price.
coupon	string	Optional	Item	"MULTIBUY_3_FOR_6"	Item-level mechanic. Independent of the order-level coupon.
index	number	Optional	Item	2	Zero-based position in the list it was shown in.
item_list_id	string	Optional	Item	"chilled__chilled-ready-meals"	Set on list events; overrides the event-level value.
item_list_name	string	Optional	Item	"Chilled — Chilled Ready Meals"	Human-readable list name.
brand_tier	string	Optional	Item	"Essentials"	CUSTOM. Essentials / Standard / Finest. Must be registered (§9).
unit_price	string	Optional	Item	"£5.00/kg"	CUSTOM. The shelf-edge comparison price.
substitutable	string	Optional	Item	"true"	CUSTOM. Only from add_payment_info onwards, when the preference is known.

Multibuy pricing: there is no mechanic field

GA4 has no structured field for "3 for £6". The documented levers are `price` (charged), `discount` (per-unit saving) and `coupon` (a string). We use all three, with the invariant `price + discount === shelfPrice` holding exactly for every promoted line. The mechanic itself goes in the item-level `coupon` as a token (`MULTIBUY_3_FOR_6`), which stays queryable without free-text parsing.

5. Quantity steppers: the delta rule

Grocery baskets are edited constantly, so the stepper is where a grocery implementation most easily goes wrong. **Google documents no rule for what a stepper should report**, and both readings are defensible. This is our decision, and it is binding across the whole site.

> **Decision: report the delta, never the new total**
> A stepper moved 2 → 5 fires `add_to_cart` with `quantity: 3`. Moved 5 → 3, it fires `remove_from_cart` with `quantity: 2`.

Three reasons:

- The events are named for *actions*. "Added to cart" describes the units just added, not the basket's new state.
- It keeps `value` honest. On the naive absolute reading, a stepper walked 1 → 2 → 3 reports £2 + £4 + £6 and add-to-cart revenue triples.
- Every worked example in Google's own documentation adds an item once with a fixed quantity, which is consistent with the delta reading.

The cost is that `add_to_cart` totals no longer reconcile to basket contents by inspection. `view_cart` is the event that reports basket state. Whichever convention a team picks, it must not be mixed — GA4 will never catch you.

STEPPER MOVED 2 → 5. GENERATED FROM THE IMPLEMENTATION.

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "add_to_cart",
  "ecommerce": {
    "currency": "GBP",
    "value": 6,
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 3,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg"
      }
    ]
  }
});
```

THE SAME STEPPER MOVED 5 → 3

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "remove_from_cart",
  "ecommerce": {
    "currency": "GBP",
    "value": 4,
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 2,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg"
      }
    ]
  }
});
```

6. Funnel order reconciliation

Real grocery journeys book a delivery slot *before* checkout. Implemented literally, that fires `add_shipping_info` before `begin_checkout` — the reverse of Google's documented order, which silently breaks GA4's built-in checkout funnel reports. Nothing errors; the funnel simply reports nonsense.

> **Resolution: move the event, not the screens**
>
> The shopper's flow is unchanged — trolley → sign in → slot → checkout. Only the event moves: `begin_checkout` fires on the **trolley's primary button**, which is the actual moment of commitment, rather than on arrival at the checkout page. The events then emit in canonical order:
>
> view_cart → begin_checkout → login → add_shipping_info → add_payment_info → purchase

Note where `login` lands: between commitment and shipping. Google's ecommerce guide never mentions `login`, so there is no documented position for it — we fire it where the UI actually gates the user, which is what makes `login_status` a meaningful funnel split (§8).

7. Payment methods, in full

`payment_type` is snake_case and drawn from a closed vocabulary. Every supported method is shown below in full, because a half-specified enum is how free text creeps into a dimension.

PAYMENT_TYPE: "CARD"

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "add_payment_info",
  "ecommerce": {
    "currency": "GBP",
    "value": 6,
    "payment_type": "card",
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 3,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "index": 0,
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg",
        "substitutable": "true"
      }
    ]
  }
});
```

PAYMENT_TYPE: "APPLE_PAY"

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "add_payment_info",
  "ecommerce": {
    "currency": "GBP",
    "value": 6,
    "payment_type": "apple_pay",
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 3,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "index": 0,
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg",
        "substitutable": "true"
      }
    ]
  }
});
```

PAYMENT_TYPE: "GOOGLE_PAY"

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "add_payment_info",
  "ecommerce": {
    "currency": "GBP",
    "value": 6,
    "payment_type": "google_pay",
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 3,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "index": 0,
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg",
        "substitutable": "true"
      }
    ]
  }
});
```

PAYMENT_TYPE: "VOUCHER"

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "add_payment_info",
  "ecommerce": {
    "currency": "GBP",
    "value": 6,
    "payment_type": "voucher",
    "coupon": "GIFTCARD",
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 3,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "index": 0,
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg",
        "substitutable": "true"
      }
    ]
  }
});
```

8. Identity: user_id and user properties

The slot-booking gate produces a genuine anonymous → identified transition mid-funnel, which is the most useful thing about it analytically.

> **user_id is a Google tag setting, not a user property**
>
> `user_id` is pushed as a **top-level dataLayer key** and read into the Google tag's `user_id` *configuration setting* in GTM. Putting it inside the `user_properties` object is a common and completely silent failure: GA4 accepts it as an ordinary user property, cross-device stitching never happens, and nothing errors.
>
> On sign-out it must be `null` — not `""`, not the string `"null"`, both of which GA4 treats as a real ID.

PUSHED BEFORE THE LOGIN EVENT, AND AGAIN ON EVERY PAGE LOAD

```
window.dataLayer.push({
  "user_id": "FC-U-C6JPYENV",
  "user_properties": {
    "login_status": "logged_in",
    "customer_type": "new",
    "delivery_pass_holder": "false"
  }
});

window.dataLayer.push({ "event": "login", "method": "email" });
```

Two details that are easy to miss on a multi-page site. First, **identity must be re-published on every page load** — each navigation is a new document with an empty dataLayer. Second, it must be published **before GTM initialises**, so the Google tag's automatic page view already carries the `user_id`. Doing it from a React effect is too late and unreliable, so it runs as an inline script between the consent defaults and the GTM loader.

The ID itself is pseudonymous and generated once per browser. **It is never an email address and never derived from one** — a client-side hash of an email is still PII and is reversible against a known address list.

User property	Values	Example	Why
login_status	logged_in / logged_out	"logged_in"	Splits the funnel at exactly the point the gate bites.
customer_type	new / returning	"new"	First order versus repeat. Browser-local here; production reads it from the CRM.
delivery_pass_holder	true / false	"false"	Pass holders pay no delivery fee, so their basket economics differ.

> **Never send PII**
>
> No email address, name, postcode or card detail is sent as a parameter or user property, in any event. The login form collects an email; it never leaves the browser.

9. Custom dimensions to register

Collecting a custom parameter is not enough. Every one must be registered in the GA4 admin (Admin → Custom definitions) with a display name, parameter name and scope, or it is collected and never appears in a single report. This is a frequent and completely invisible failure — the data is in the hit, and the report is empty.

Parameter	Scope	Display name	Notes
slot_day	Event	Delivery slot day	ISO date
slot_window	Event	Delivery slot window	e.g. 18:00 - 19:00
slot_fee	Event	Delivery slot fee	GBP; 0 for pass holders
search_results_count	Event	Search results count	Makes zero-result searches visible
content_type	Event	Content type	department_tile / aisle_tile / product_sort
content_id	Event	Content ID	The slug or sort key
consent_action	Event	Consent action	accept / reject / customise
substitutions_allowed	Event	Substitutions allowed	Boolean
basket_value	Event	Basket value	On minimum_spend_not_met
shortfall	Event	Minimum spend shortfall	Pre-computed
login_status	User	Login status	logged_in / logged_out
customer_type	User	Customer type	new / returning
delivery_pass_holder	User	Delivery Pass holder	true / false
brand_tier	Item	Brand tier	Essentials / Standard / Finest
unit_price	Item	Unit price	Shelf-edge comparison price
substitutable	Item	Substitutable	true / false

> **Limits, from Google's documentation**
>
> - 25 user properties per property.
> - Parameter names: user-scoped ≤ 24 characters, event-scoped ≤ 40.
> - User property *values* ≤ 36 characters; `user_id` ≤ 256.
> - Item-scoped custom dimensions: 10 on a Standard property, 25 on 360. We use 3.

10. Consent Mode v2

Defaults are set before GTM loads, in Freshcart's own layout. **That ordering is the entire point** — a default set after a tag has already read or written a cookie is meaningless, and is one of Tag Assistant's named failure modes.

INLINE SCRIPT, EXECUTED BEFORE THE GTM LOADER

```
window.dataLayer = window.dataLayer || [];
function gtag(){dataLayer.push(arguments);}

gtag('consent', 'default', {
  'ad_storage': 'denied',
  'ad_user_data': 'denied',
  'ad_personalization': 'denied',
  'analytics_storage': 'denied',
  'functionality_storage': 'granted',
  'personalization_storage': 'denied',
  'security_storage': 'granted',
  'wait_for_update': 500
});

gtag('set', 'url_passthrough', true);
gtag('set', 'ads_data_redaction', true);
```

`wait_for_update: 500` gives the banner half a second to answer before tags give up waiting and fire in the denied state. `url_passthrough` propagates ad-click IDs across same-domain links without cookies; `ads_data_redaction` strips them from the network request itself when `ad_storage` is denied.

The update, and why it is not an array

SRC/LIB/FRESHCART/DATALAYER.TS → SRC/LIB/CONSENT.TS

```
function gtag() {
  window.dataLayer.push(arguments);   // arguments, NOT an array
}
gtag('consent', 'update', preferences);
```

> **This is the subtlest failure in the whole implementation**
>
> A gtag command is only recognised when the pushed value is an **arguments object**. Pushing `["consent", "update", prefs]` instead looks identical — ordered, has a `length` — succeeds, appears correctly in the dataLayer and in any debugger overlay, raises no error, and **does absolutely nothing**.
>
> This shipped broken on the sibling site and cost real consent data. The only visible symptom was that hits kept reporting `gcs=G100` after the shopper pressed Accept all. Note the diagnostic asymmetry: the *restore* path reads saved consent on page load using a correct `gtag()` call, so reloading made it look like it worked.
>
> Inside a GTM *custom template* the answer differs again: use Tag Manager's sandboxed `updateConsentState` API rather than a gtag command.

What changes under denial

Signal denied	Effect
analytics_storage	No GA4 cookies. Events still send as cookieless pings, with no client ID and no session continuity. Aggregate counts survive; user-level reporting does not.
ad_storage	No advertising cookies. Google Ads conversions send to a cookieless domain; remarketing is blocked outright.
ad_user_data	No user data sent to Google for advertising. Enhanced conversions stop matching.
ad_personalization	No personalised advertising; remarketing audiences are not built.

A cookieless ping still carries the event name, page location, timestamp and consent state; it carries no client ID, no user ID and no advertising identifiers. GA4 uses them for behavioural and conversion modelling rather than discarding them. Consent state travels on every hit as the `gcs` parameter — `G100` is both denied, `G111` both granted, and it is the fastest way to verify a consent implementation from the network tab.

> **Google Ads Remarketing does not support URL passthrough**
>
> Unlike Conversion and Floodlight tags, the Remarketing tag has no URL-passthrough behaviour: under denied `ad_storage` its requests and cookies are blocked entirely. Do not expect denied-state remarketing coverage.

11. Enhanced Measurement: what not to implement

Deciding what *not* to build is as much a part of a measurement plan as deciding what to build. Enhanced Measurement already collects several of the events this site needs. Implementing them again does not produce better data — it produces double data.

Event	Source	Do we implement it?
page_view	Google tag, send_page_view (default true)	**No — every navigation is a real page load**
view_search_results	Enhanced Measurement, Site search	**No — our results URL uses ?q=**
click (outbound)	Enhanced Measurement	No
file_download	Enhanced Measurement	No — covers the spec PDF link
scroll	Enhanced Measurement	No
search	Our code	Yes — adds search_results_count

Site search: the same trap, from the other direction

Enhanced Measurement's Site search fires `view_search_results` automatically whenever a URL carries one of its default query parameters — `q`, `s`, `search`, `query`, `keyword`. Our results page is `/freshcart/search?q=…`, so it is already covered, and pushing `view_search_results` ourselves would double every search.

We *do* push `search`, because it adds something Enhanced Measurement cannot: `search_results_count`. A zero-result search is one of the highest-value merchandising signals in grocery — it is a shopper telling you what your catalogue is missing — and it is invisible without that parameter. The two events are not duplicates: they measure the query and the results view respectively.

This is the same class of error as a manual `page_view` on this site, approached from the opposite side. In both cases the discipline is identical: establish what the platform already sends before writing a line of tracking code.

12. Event delivery and navigation

On a multi-page site, pushing an event and immediately navigating races the document teardown. There are two answers, and using the heavier one everywhere is a mistake.

Case	Approach	Why
Ordinary link clicks — `select_item`, `select_promotion`, `select_content`	Push and let the navigation proceed	GA4 tags send via navigator.sendBeacon, which is built to survive unload. Intercepting every link would add visible delay and break middle-click and cmd-click — a bad trade for an event that almost always gets out.
The checkout commitment — `begin_checkout`, `add_shipping_info`, `login`, `sign_up`	GTM's `eventCallback` + `eventTimeout`, plus a timeout fallback	These are buttons, not links (no new-tab semantics to preserve), they happen once per journey, and losing one puts a hole in the middle of the funnel.

The fallback timer is not optional

`eventCallback` only fires if GTM runs. Blocked by an extension, offline, or held by denied consent, it never arrives — and without an independent timeout the shopper clicks Checkout and simply stays put. Measurement must never be able to break the shop.

13. Event reference

Every event, with the precise trigger, its event-level parameters, edge cases, and — where it adds something — the generated payload. The `items[]` schema is documented once in §4 rather than repeated.

view_item_list GA4 recommended event Browse

When it fires. On render of an aisle page (/freshcart/browse/[department]/[aisle]), the search results page, and the home page's offers row. Re-fires when the sort order changes, because re-sorting changes which product was impressed at each index.

Parameter	Type	Required	Scope	Example	Notes
item_list_id	string	Optional	Event	"fruit-veg__fresh-fruit"	Stable machine ID for the list. `department__aisle` on aisle pages.
item_list_name	string	Optional	Event	"Fruit & Veg — Fresh Fruit"	Human-readable list name, as shown to the shopper.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.

AISLE PAGE: FRUIT & VEG → FRESH FRUIT, THREE PRODUCTS

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "view_item_list",
  "ecommerce": {
    "item_list_id": "fruit-veg__fresh-fruit",
    "item_list_name": "Fruit & Veg — Fresh Fruit",
    "items": [
      {
        "item_id": "fc-0001",
        "item_name": "Freshcart British Gala Apples",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Fruit & Veg",
        "item_category3": "Fresh Fruit",
        "item_category4": "Apples & Pears",
        "item_variant": "6 pack",
        "price": 1.85,
        "quantity": 1,
        "index": 0,
        "item_list_id": "fruit-veg__fresh-fruit",
        "item_list_name": "Fruit & Veg — Fresh Fruit",
        "brand_tier": "Standard",
        "unit_price": "31p each"
      },
      {
        "item_id": "fc-0002",
        "item_name": "Freshcart Essentials Conference Pears",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Fruit & Veg",
        "item_category3": "Fresh Fruit",
        "item_category4": "Apples & Pears",
        "item_variant": "5 pack",
        "price": 0.95,
        "quantity": 1,
        "index": 1,
        "item_list_id": "fruit-veg__fresh-fruit",
        "item_list_name": "Fruit & Veg — Fresh Fruit",
        "brand_tier": "Essentials",
        "unit_price": "19p each"
      },
      {
        "item_id": "fc-0003",
        "item_name": "Grove & Vale Blueberries",
        "item_brand": "Grove & Vale",
        "item_category": "Fresh Food",
        "item_category2": "Fruit & Veg",
        "item_category3": "Fresh Fruit",
        "item_category4": "Berries",
        "item_variant": "200g",
        "price": 2.75,
        "quantity": 1,
        "discount": 0.5,
        "coupon": "ROLLBACK",
        "index": 2,
        "item_list_id": "fruit-veg__fresh-fruit",
        "item_list_name": "Fruit & Veg — Fresh Fruit",
        "brand_tier": "Finest",
        "unit_price": "£13.75/kg"
      }
    ]
  }
});
```

Edge cases

- Carries no `currency` or `value`. GA4 requires `currency` only when `value` is set, and a list impression is not worth the sum of everything in it.
- `index` is zero-based and must match the on-screen order. It is what attributes a later `select_item` back to its list position.
- Not fired on the department page — that page lists aisles, not products, so there is no `items[]` to populate.
- Not fired when a search returns zero results; only `search` fires, carrying `search_results_count: 0`.

select_item GA4 recommended event Browse

When it fires. Click of a product tile on any list — aisle page, search results, or the home offers row.

Parameter	Type	Required	Scope	Example	Notes
item_list_id	string	Optional	Event	"fruit-veg__fresh-fruit"	Stable machine ID for the list. `department__aisle` on aisle pages.
item_list_name	string	Optional	Event	"Fruit & Veg — Fresh Fruit"	Human-readable list name, as shown to the shopper.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.

THIRD TILE IN THE LIST CLICKED (INDEX 2)

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "select_item",
  "ecommerce": {
    "item_list_id": "fruit-veg__fresh-fruit",
    "item_list_name": "Fruit & Veg — Fresh Fruit",
    "items": [
      {
        "item_id": "fc-0003",
        "item_name": "Grove & Vale Blueberries",
        "item_brand": "Grove & Vale",
        "item_category": "Fresh Food",
        "item_category2": "Fruit & Veg",
        "item_category3": "Fresh Fruit",
        "item_category4": "Berries",
        "item_variant": "200g",
        "price": 2.75,
        "quantity": 1,
        "discount": 0.5,
        "coupon": "ROLLBACK",
        "index": 2,
        "item_list_id": "fruit-veg__fresh-fruit",
        "item_list_name": "Fruit & Veg — Fresh Fruit",
        "brand_tier": "Finest",
        "unit_price": "£13.75/kg"
      }
    ]
  }
});
```

Edge cases

- No `currency`/`value`, per the GA4 reference.
- `index`, `item_list_id` and `item_list_name` must be identical to the values used by the `view_item_list` that produced this list, or attribution breaks.
- Fires on a link click that immediately navigates. We rely on the GA4 tag's `sendBeacon` transport rather than delaying the navigation — see §12.

view_item GA4 recommended event Browse

When it fires. On render of a product detail page, /freshcart/product/[id].

Parameter	Type	Required	Scope	Example	Notes
currency	string	Conditional	Event	"GBP"	Required whenever `value` is set. ISO 4217. Always GBP here.
value	number	Conditional	Event	11.65	Σ(price × quantity) across `items`. Never includes shipping or tax. Example is for the worked basket in §13.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.

PDP FOR A PROMOTED LINE (3 FOR £6)

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "view_item",
  "ecommerce": {
    "currency": "GBP",
    "value": 2,
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 1,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg"
      }
    ]
  }
});
```

Edge cases

- `value` is the unit price actually charged — the discounted price for a promoted line, not the shelf price.
- Guarded by a `useRef` keyed on the product ID, so a re-render cannot double-count.
- Fires once per document. Freshcart is a multi-page app, so revisiting the same PDP is a new page load and a legitimate new `view_item`.

add_to_cart GA4 recommended event Basket

When it fires. The 'Add' button, or any INCREASE of a quantity stepper, anywhere in the site — aisle tile, PDP, or the trolley itself.

Parameter	Type	Required	Scope	Example	Notes
currency	string	Conditional	Event	"GBP"	Required whenever `value` is set. ISO 4217. Always GBP here.
value	number	Conditional	Event	11.65	Σ(price × quantity) across `items`. Never includes shipping or tax. Example is for the worked basket in §13.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.

STEPPER MOVED 2 → 5: REPORTS THE DELTA OF 3

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "add_to_cart",
  "ecommerce": {
    "currency": "GBP",
    "value": 6,
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 3,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg"
      }
    ]
  }
});
```

Edge cases

- DELTA, NOT TOTAL. A stepper moved 2 → 5 fires with `quantity: 3`. See §5 for the full reasoning.
- `value` is `price × delta`, so add-to-cart revenue sums correctly over a session.
- Quantity is clamped to 20 per line; the event always reports what actually landed in the trolley, never the attempted value.
- A no-op change (delta of 0) fires nothing at all.

remove_from_cart GA4 recommended event Basket

When it fires. Any DECREASE of a quantity stepper, including removing a line entirely (which is a decrease to zero).

Parameter	Type	Required	Scope	Example	Notes
currency	string	Conditional	Event	"GBP"	Required whenever `value` is set. ISO 4217. Always GBP here.
value	number	Conditional	Event	11.65	Σ(price × quantity) across `items`. Never includes shipping or tax. Example is for the worked basket in §13.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.

STEPPER MOVED 5 → 3: REPORTS 2 REMOVED

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "remove_from_cart",
  "ecommerce": {
    "currency": "GBP",
    "value": 4,
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 2,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg"
      }
    ]
  }
});
```

Edge cases

- Delta again: 5 → 3 reports `quantity: 2`, not 3.
- Removing a line of 4 fires once with `quantity: 4`, not four separate events.

view_cart GA4 recommended event Basket

When it fires. On render of the trolley page, /freshcart/trolley.

Parameter	Type	Required	Scope	Example	Notes
currency	string	Conditional	Event	"GBP"	Required whenever `value` is set. ISO 4217. Always GBP here.
value	number	Conditional	Event	11.65	Σ(price × quantity) across `items`. Never includes shipping or tax. Example is for the worked basket in §13.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.

THREE LINES AT MIXED QUANTITIES, MIXING PROMOTED AND FULL-PRICE

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "view_cart",
  "ecommerce": {
    "currency": "GBP",
    "value": 11.65,
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 3,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "index": 0,
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg"
      },
      {
        "item_id": "fc-0011",
        "item_name": "Freshcart British Semi-Skimmed Milk",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Milk & Cream",
        "item_category4": "Fresh Milk",
        "item_variant": "2L",
        "price": 1.45,
        "quantity": 2,
        "index": 1,
        "brand_tier": "Standard",
        "unit_price": "73p/litre"
      },
      {
        "item_id": "fc-0003",
        "item_name": "Grove & Vale Blueberries",
        "item_brand": "Grove & Vale",
        "item_category": "Fresh Food",
        "item_category2": "Fruit & Veg",
        "item_category3": "Fresh Fruit",
        "item_category4": "Berries",
        "item_variant": "200g",
        "price": 2.75,
        "quantity": 1,
        "discount": 0.5,
        "coupon": "ROLLBACK",
        "index": 2,
        "brand_tier": "Finest",
        "unit_price": "£13.75/kg"
      }
    ]
  }
});
```

Edge cases

- Fires for an EMPTY trolley too, with `value: 0` and `items: []`. An empty basket view is a real funnel event and suppressing it hides abandonment.
- This is the event that reports basket STATE. `add_to_cart` reports actions; do not try to reconcile the two by summing.
- Must not fire until client storage has been read — see the hydration trap in §14.

begin_checkout

GA4 recommended event

Checkout

When it fires. Click of the trolley's primary button — 'Book a slot' or 'Checkout'. NOT on arrival at /freshcart/checkout.

Parameter	Type	Required	Scope	Example	Notes
currency	string	Conditional	Event	"GBP"	Required whenever `value` is set. ISO 4217. Always GBP here.
value	number	Conditional	Event	11.65	Σ(price × quantity) across `items`. Never includes shipping or tax. Example is for the worked basket in §13.
coupon	string	Optional	Event	"FRESH10"	Order-level voucher. Independent of the item-level `coupon` carrying a multibuy mechanic.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.

COMMITTING TO CHECKOUT WITH A VOUCHER ALREADY ENTERED

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "begin_checkout",
  "ecommerce": {
    "currency": "GBP",
    "value": 11.65,
    "coupon": "FRESH10",
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 3,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "index": 0,
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg"
      },
      {
        "item_id": "fc-0011",
        "item_name": "Freshcart British Semi-Skimmed Milk",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Milk & Cream",
        "item_category4": "Fresh Milk",
        "item_variant": "2L",
        "price": 1.45,
        "quantity": 2,
        "index": 1,
        "brand_tier": "Standard",
        "unit_price": "73p/litre"
      },
      {
        "item_id": "fc-0003",
        "item_name": "Grove & Vale Blueberries",
        "item_brand": "Grove & Vale",
        "item_category": "Fresh Food",
        "item_category2": "Fruit & Veg",
        "item_category3": "Fresh Fruit",
        "item_category4": "Berries",
        "item_variant": "200g",
        "price": 2.75,
        "quantity": 1,
        "discount": 0.5,
        "coupon": "ROLLBACK",
        "index": 2,
        "brand_tier": "Finest",
        "unit_price": "£13.75/kg"
      }
    ]
  }
});
```

Edge cases

- Placed at the trolley → slot transition deliberately, to keep events in canonical order. See §6.
- Fires on a button, so navigation is deferred until GTM confirms the event was processed (`eventCallback`), with a timeout fallback if GTM never loads.
- Fires on every genuine checkout attempt. A shopper who returns to the trolley and commits again produces a second `begin_checkout`, which is correct — GA4 funnels count users, not events.

add_shipping_info GA4 recommended event Checkout

When it fires. Confirming a delivery slot on /freshcart/slots ('Confirm slot and continue').

Parameter	Type	Required	Scope	Example	Notes
currency	string	Conditional	Event	"GBP"	Required whenever `value` is set. ISO 4217. Always GBP here.
value	number	Conditional	Event	11.65	Σ(price × quantity) across `items`. Never includes shipping or tax. Example is for the worked basket in §13.
shipping_tier	string	Optional	Event	"Peak Evening"	Bucketed, not the raw slot. One of "Standard Midweek", "Peak Evening", "Peak Weekend", "Express 4-Hour".
coupon	string	Optional	Event	"FRESH10"	Order-level voucher. Independent of the item-level `coupon` carrying a multibuy mechanic.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.
slot_day	string	Optional	Event	"2026-09-02"	ISO date. Register as a custom dimension to report on it.
slot_window	string	Optional	Event	"18:00 - 19:00"	The delivery window as shown.
slot_fee	number	Optional	Event	4.5	Fee for this slot, in GBP. 0 for Delivery Pass holders.

PEAK EVENING SLOT, £4.50, WITH THE CUSTOM SLOT PARAMETERS

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "add_shipping_info",
  "ecommerce": {
    "currency": "GBP",
    "value": 11.65,
    "shipping_tier": "Peak Evening",
    "coupon": "FRESH10",
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 3,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "index": 0,
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg"
      },
      {
        "item_id": "fc-0011",
        "item_name": "Freshcart British Semi-Skimmed Milk",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Milk & Cream",
        "item_category4": "Fresh Milk",
        "item_variant": "2L",
        "price": 1.45,
        "quantity": 2,
        "index": 1,
        "brand_tier": "Standard",
        "unit_price": "73p/litre"
      },
      {
        "item_id": "fc-0003",
        "item_name": "Grove & Vale Blueberries",
        "item_brand": "Grove & Vale",
        "item_category": "Fresh Food",
        "item_category2": "Fruit & Veg",
        "item_category3": "Fresh Fruit",
        "item_category4": "Berries",
        "item_variant": "200g",
        "price": 2.75,
        "quantity": 1,
        "discount": 0.5,
        "coupon": "ROLLBACK",
        "index": 2,
        "brand_tier": "Finest",
        "unit_price": "£13.75/kg"
      }
    ]
  },
  "slot_day": "2026-09-02",
  "slot_window": "18:00 - 19:00",
  "slot_fee": 4.5
});
```

Edge cases

- `shipping_tier` is a LABEL and exists only on this event. `shipping` is a COST and exists only on `purchase`. They are different parameters; do not conflate them.
- The slot fee is NOT sent as a cost here — only as the custom `slot_fee` parameter. The cost reaches GA4 at `purchase`, in `shipping`.
- For a Delivery Pass holder the fee is waived, so `slot_fee` is 0 and the stored slot matches. Reporting the headline price while charging zero is how `slot_fee` ends up disagreeing with the receipt.
- Bucketing is a deliberate choice: a composite string like "Wed 2 Sep 18:00-19:00" is inside the documented schema but cannot be pivoted in GA4 without regex.

add_payment_info GA4 recommended event Checkout

When it fires. Submitting the payment step ('Place order') on /freshcart/checkout.

Parameter	Type	Required	Scope	Example	Notes
currency	string	Conditional	Event	"GBP"	Required whenever `value` is set. ISO 4217. Always GBP here.
value	number	Conditional	Event	11.65	Σ(price × quantity) across `items`. Never includes shipping or tax. Example is for the worked basket in §13.
payment_type	string	Optional	Event	"apple_pay"	One of `card`, `apple_pay`, `google_pay`, `voucher`. snake_case.
coupon	string	Optional	Event	"FRESH10"	Order-level voucher. Independent of the item-level `coupon` carrying a multibuy mechanic.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.

Edge cases

- From this event onwards, items carry the item-scoped `substitutable` flag — the order-level substitution preference is only known at checkout.
- See §7 for the payload of every payment method.
- The 'placing your order' pause that follows comfortably outlasts the push, so the event never races the navigation.

purchase GA4 recommended event Checkout

When it fires. On render of /freshcart/confirmation, exactly once per order number.

Parameter	Type	Required	Scope	Example	Notes
transaction_id	string	Required	Event	"FC-2026090201-8KQ4"	The de-duplication key. GA4 and Google Ads both use it to discard a repeated purchase.
currency	string	Conditional	Event	"GBP"	Required whenever `value` is set. ISO 4217. Always GBP here.
value	number	Conditional	Event	11.65	Σ(price × quantity) across `items`. Never includes shipping or tax. Example is for the worked basket in §13.
shipping	number	Optional	Event	8.5	Delivery fee PLUS any minimum-basket charge. A cost, not a tier.
tax	number	Optional	Event	0	Zero here — most UK grocery food is zero-rated.
coupon	string	Optional	Event	"FRESH10"	Order-level voucher. Independent of the item-level `coupon` carrying a multibuy mechanic.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.
payment_type	string	Optional	Event	"card"	Repeated on `purchase` so the method is queryable per order.
slot_day	string	Optional	Event	"2026-09-02"	ISO date. Register as a custom dimension to report on it.
slot_window	string	Optional	Event	"18:00 - 19:00"	The delivery window as shown.
slot_fee	number	Optional	Event	4.5	Fee for this slot, in GBP. 0 for Delivery Pass holders.

MULTIPLE ITEMS, ORDER-LEVEL VOUCHER, ITEM-LEVEL MULTIBUY, SHIPPING = DELIVERY FEE + MINIMUM-BASKET CHARGE

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "purchase",
  "ecommerce": {
    "transaction_id": "FC-2026090201-8KQ4",
    "currency": "GBP",
    "value": 11.65,
    "shipping": 8.5,
    "tax": 0,
    "coupon": "FRESH10",
    "items": [
      {
        "item_id": "fc-0015",
        "item_name": "Freshcart Chicken Tikka Masala",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Chilled Ready Meals",
        "item_category4": "Ready Meals",
        "item_variant": "400g",
        "price": 2,
        "quantity": 3,
        "discount": 0.5,
        "coupon": "MULTIBUY_3_FOR_6",
        "index": 0,
        "brand_tier": "Standard",
        "unit_price": "£5.00/kg",
        "substitutable": "true"
      },
      {
        "item_id": "fc-0011",
        "item_name": "Freshcart British Semi-Skimmed Milk",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Milk & Cream",
        "item_category4": "Fresh Milk",
        "item_variant": "2L",
        "price": 1.45,
        "quantity": 2,
        "index": 1,
        "brand_tier": "Standard",
        "unit_price": "73p/litre",
        "substitutable": "true"
      },
      {
        "item_id": "fc-0003",
        "item_name": "Grove & Vale Blueberries",
        "item_brand": "Grove & Vale",
        "item_category": "Fresh Food",
        "item_category2": "Fruit & Veg",
        "item_category3": "Fresh Fruit",
        "item_category4": "Berries",
        "item_variant": "200g",
        "price": 2.75,
        "quantity": 1,
        "discount": 0.5,
        "coupon": "ROLLBACK",
        "index": 2,
        "brand_tier": "Finest",
        "unit_price": "£13.75/kg",
        "substitutable": "true"
      }
    ]
  },
  "payment_type": "card",
  "slot_day": "2026-09-02",
  "slot_window": "18:00 - 19:00",
  "slot_fee": 4.5,
  "shipping_tier": "Peak Evening"
});
```

Edge cases

- `value` is goods only. Adding shipping into it inflates reported product revenue — the single most common ecommerce reporting error.

- `shipping` carries the delivery fee and the £4 minimum-basket charge combined, because both are delivery costs the shopper actually paid.
- Fires exactly once per order. Guarded by a `useRef` (same document) AND `sessionStorage` keyed on the transaction ID (survives a refresh, and still allows a genuinely different second order in the same session).
- The order is frozen to storage before the confirmation page loads, so the event reports what was placed rather than re-deriving it from a trolley that is about to be cleared.
- The trolley is cleared only AFTER the event fires.

---

view_promotion

GA4 recommended event · Promotions

When it fires. On render of the home page, for the hero offer banner and the products it promotes.

Parameter	Type	Required	Scope	Example	Notes
promotion_id	string	Optional	Event	"promo_home_hero_ready_meals"	Stable ID for the promotion itself.
promotion_name	string	Optional	Event	"3 for £6 on chilled ready meals"	The mechanic as worded to the shopper.
creative_name	string	Optional	Event	"home_hero_banner"	Which creative was shown.
creative_slot	string	Optional	Event	"home_hero"	Where on the page it appeared.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.

HOME PAGE HERO BANNER IMPRESSION

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "view_promotion",
  "ecommerce": {
    "promotion_id": "promo_home_hero_ready_meals",
    "promotion_name": "3 for £6 on chilled ready meals",
    "creative_name": "home_hero_banner",
    "creative_slot": "home_hero",
    "items": [
      {
        "item_id": "fc-0003",
        "item_name": "Grove & Vale Blueberries",
        "item_brand": "Grove & Vale",
        "item_category": "Fresh Food",
        "item_category2": "Fruit & Veg",
        "item_category3": "Fresh Fruit",
        "item_category4": "Berries",
        "item_variant": "200g",
        "price": 2.75,
        "quantity": 1,
        "discount": 0.5,
        "coupon": "ROLLBACK",
        "index": 0,
        "promotion_id": "promo_home_hero_ready_meals",
        "promotion_name": "3 for £6 on chilled ready meals",
        "brand_tier": "Finest",
        "unit_price": "£13.75/kg"
      }
    ]
  }
});
```

Edge cases

- Items carry `promotion_id`/`promotion_name` as well as the event, so a promoted item stays attributable if it appears in a later event.
- One-shot guarded — an impression is per page load.

select_promotion GA4 recommended event Promotions

When it fires. Click of the home page offer banner.

Parameter	Type	Required	Scope	Example	Notes
promotion_id	string	Optional	Event	"promo_home_hero_ready_meals"	Stable ID for the promotion itself.
promotion_name	string	Optional	Event	"3 for £6 on chilled ready meals"	The mechanic as worded to the shopper.
creative_name	string	Optional	Event	"home_hero_banner"	Which creative was shown.
creative_slot	string	Optional	Event	"home_hero"	Where on the page it appeared.
items	Array<Item>	Required	Event	[…]	See §4 for the full item schema. Max 200 items; extras are dropped.

OFFER BANNER CLICKED

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "select_promotion",
  "ecommerce": {
    "promotion_id": "promo_home_hero_ready_meals",
    "promotion_name": "3 for £6 on chilled ready meals",
    "creative_name": "home_hero_banner",
    "creative_slot": "home_hero",
    "items": [
      {
        "item_id": "fc-0003",
        "item_name": "Grove & Vale Blueberries",
        "item_brand": "Grove & Vale",
        "item_category": "Fresh Food",
        "item_category2": "Fruit & Veg",
        "item_category3": "Fresh Fruit",
        "item_category4": "Berries",
        "item_variant": "200g",
        "price": 2.75,
        "quantity": 1,
        "discount": 0.5,
        "coupon": "ROLLBACK",
        "index": 0,
        "promotion_id": "promo_home_hero_ready_meals",
        "promotion_name": "3 for £6 on chilled ready meals",
        "brand_tier": "Finest",
        "unit_price": "£13.75/kg"
      }
    ]
  }
});
```

Edge cases

- `creative_name` and `creative_slot` must match the `view_promotion` that preceded it.

login GA4 recommended event Identity

When it fires. Successful sign-in at the gate, /freshcart/login. The gate is reached by attempting to book a delivery slot.

Parameter	Type	Required	Scope	Example	Notes
method	string	Optional	Event	"email"	The only parameter these events take.

Edge cases

- The identity push (`user_id` + `user_properties`) MUST precede this event, or GTM reads the signed-out values and the event marking the transition is itself attributed to an anonymous user. See §8.
- No email address, name or password is ever sent.

sign_up GA4 recommended event Identity

When it fires. Successful registration at the gate, using the Register tab of the same form.

Parameter	Type	Required	Scope	Example	Notes
method	string	Optional	Event	"email"	Same shape as `login`.

Edge cases

- Distinct from `login` on purpose — new-account rate at the gate is a different KPI from returning sign-in rate.

search GA4 recommended event Engagement

When it fires. On render of /freshcart/search?q=..., once per distinct search term.

Parameter	Type	Required	Scope	Example	Notes
search_term	string	Optional	Event	"semi skimmed milk"	Exactly what the shopper typed, trimmed.
search_results_count	number	Optional	Event	2	Custom parameter. Register it to report zero-result searches, which Enhanced Measurement cannot give you.

A SEARCH RETURNING TWO PRODUCTS

```
window.dataLayer.push({
  "event": "search",
  "search_term": "semi skimmed milk",
  "search_results_count": 2
});
```

Edge cases

- Fired on the RESULTS page, not on submit. Submitting tears the document down immediately, and a push fired on submit races the unload.
- We deliberately do NOT push `view_search_results` — Enhanced Measurement already sends it. See §11.
- Not fired for an empty query.

select_content GA4 recommended event Engagement

When it fires. Department tile clicks on the home page, aisle tile clicks on a department page, and changes to the aisle sort control.

Parameter	Type	Required	Scope	Example	Notes
content_type	string	Optional	Event	"product_sort"	One of `department_tile`, `aisle_tile`, `product_sort`. snake_case.
content_id	string	Optional	Event	"price_low_high"	The slug or sort key selected.

AISLE SORT CHANGED

```
window.dataLayer.push({
  "event": "select_content",
  "content_type": "product_sort",
  "content_id": "price_low_high"
});
```

Edge cases

- A recommended event is preferred over a bespoke `sort_changed`: recommended events unlock built-in reporting, custom ones do not.
- `content_type` is the discriminator — keep the vocabulary closed and documented, or it becomes unqueryable.

slot_selected Custom event Grocery-specific

When it fires. Clicking a delivery window in the slot grid — BEFORE it is confirmed.

Parameter	Type	Required	Scope	Example	Notes
slot_day	string	Optional	Event	"2026-09-02"	ISO date. Register as a custom dimension to report on it.
slot_window	string	Optional	Event	"18:00 - 19:00"	The delivery window as shown.
slot_fee	number	Optional	Event	4.5	Fee for this slot, in GBP. 0 for Delivery Pass holders.
shipping_tier	string	Optional	Event	"Peak Evening"	Same bucketing as `add_shipping_info`.

A PEAK EVENING WINDOW SELECTED

```
window.dataLayer.push({
  "event": "slot_selected",
  "slot_day": "2026-09-02",
  "slot_window": "18:00 – 19:00",
  "slot_fee": 4.5,
  "shipping_tier": "Peak Evening"
});
```

Edge cases

- Deliberately separate from `add_shipping_info`: this measures slot SHOPPING (how many windows a shopper tries before committing, and which sell out), not slot commitment.
- Fires on every selection, including changing your mind. That repetition is the signal.

substitution_preference_changed Custom event Grocery-specific

When it fires. Toggling the substitutions checkbox on the order review step.

Parameter	Type	Required	Scope	Example	Notes
substitutions_allowed	boolean	Optional	Event	false	The NEW state after the toggle.

SHOPPER OPTS OUT OF SUBSTITUTIONS

```
window.dataLayer.push({
  "event": "substitution_preference_changed",
  "substitutions_allowed": false
});
```

Edge cases

- Substitutions are allowed by default, so this event overwhelmingly means an opt-OUT. That asymmetry is the point.
- The resulting preference also rides on items as the item-scoped `substitutable` flag from `add_payment_info` onwards.

minimum_spend_not_met Custom event Grocery-specific

When it fires. On render of the trolley when the subtotal is above zero but below the £40 delivery minimum.

Parameter	Type	Required	Scope	Example	Notes
basket_value	number	Optional	Event	11.65	Current subtotal.
minimum_spend	number	Optional	Event	40	The threshold in force.
shortfall	number	Optional	Event	28.35	Pre-computed so reports need no derived metric.

BASKET SHORT OF THE £40 MINIMUM

```
window.dataLayer.push({
  "event": "minimum_spend_not_met",
  "basket_value": 11.65,
  "minimum_spend": 40,
  "shortfall": 28.35
});
```

Edge cases

- Not fired for an empty trolley — a shopper who has added nothing has not failed to reach a minimum.
- Fires alongside `view_cart`, not instead of it.

consent_choice Custom event Consent

When it fires. Pressing Accept all, Reject all, or Save preferences on the consent banner.

Parameter	Type	Required	Scope	Example	Notes
consent_action	string	Optional	Event	"customise"	One of `accept`, `reject`, `customise`. British spelling — deliberate, and a GTM trigger or report keyed on it breaks if "americanized".

Edge cases

- Measures the consent RATE. It does not itself change consent — that is the `consent`/`update` command, which is a separate mechanism entirely (§10).
- Must have a matching GTM trigger. Roamio's did not, and consent-rate data never reached GA4 for weeks.

consent_update Custom event Consent

When it fires. Immediately after a consent decision is applied, carrying the resulting state as flat parameters.

Parameter	Type	Required	Scope	Example	Notes
analytics_storage	"granted" | "denied"	Optional	Event	"granted"	Flat, not nested — nested consent state was a past bug.
ad_storage	"granted" | "denied"	Optional	Event	"granted"	Plus `ad_user_data` and `ad_personalization`, same shape.

Edge cases

- Informational only. It reports the decision; the `consent`/`update` command enacts it, and is pushed FIRST.
- Ordering matters: pushed the other way round, the event announcing the decision can be blocked by the very state it announces.

purchase without a promotion or voucher

The contrast case: no item carries discount or an item-level coupon, there is no order-level coupon, and the basket clears the £40 minimum so shipping is the delivery fee alone.

A PLAIN PURCHASE, FOR COMPARISON WITH §13 PURCHASE

```
window.dataLayer.push({
  "ecommerce": null
});

window.dataLayer.push({
  "event": "purchase",
  "ecommerce": {
    "transaction_id": "FC-2026090202-2ZR7",
    "currency": "GBP",
    "value": 2.9,
    "shipping": 4.5,
    "tax": 0,
    "items": [
      {
        "item_id": "fc-0011",
        "item_name": "Freshcart British Semi-Skimmed Milk",
        "item_brand": "Freshcart",
        "item_category": "Fresh Food",
        "item_category2": "Chilled",
        "item_category3": "Milk & Cream",
        "item_category4": "Fresh Milk",
        "item_variant": "2L",
        "price": 1.45,
        "quantity": 2,
        "index": 0,
        "brand_tier": "Standard",
        "unit_price": "73p/litre",
        "substitutable": "true"
      }
    ]
  },
  "payment_type": "card",
  "slot_day": "2026-09-02",
  "slot_window": "18:00 – 19:00",
  "slot_fee": 4.5,
  "shipping_tier": "Peak Evening"
});
```

14. QA and validation

Per-event verification

1. **GTM Preview.** Connect Tag Assistant to the site and walk the funnel. For each event confirm it appears in the left-hand timeline, that the expected tag is under *Tags Fired* (not *Tags Not Fired*), and that the `ecommerce` object in the Data Layer tab matches §13.
2. **GA4 DebugView.** With Preview connected, confirm each event arrives, and inspect parameters. Check `value`, `currency`, `transaction_id` and the item count specifically.
3. **The `gcs` parameter.** On any outgoing hit, `gcs=G100` means both storages denied and `G111` means both granted. Accept the banner and confirm a subsequent hit flips to `G111` *without reloading* — reloading takes the restore path, which can pass while the banner is broken.

The check that catches silent data loss

A dataLayer push with no matching GTM trigger looks *identical* in the browser to one that fired a tag. The console shows it, the debugger overlay shows it, nothing errors — and it reaches nothing. The sibling site shipped four events that way, including `view_cart`, and it went unnoticed because the code was demonstrably “working”.

So before release, diff the event names the code pushes against the container's trigger. The code exports its own list (`FRESHCART_EVENT_NAMES`) precisely so this can be mechanical:

THE TRIGGER REGEX MUST MATCH THIS LIST EXACTLY

```
^(view_item_list|select_item|view_item|add_to_cart|remove_from_cart|view_cart|begin_checkout|add_shipping_info|add_payment_info|purchase|view_promotion|select_promotion|login|sign_up|search|select_content|slot_selected|substitution_preference_changed|minimum_spend_not_met|consent_choice|consent_update)$
```

Why manual verification is not optional

Two defects in this implementation passed lint, TypeScript and a production build, and were found only by walking the funnel in a real browser. Both are worth stating, because both are generic to this architecture.

> **1. view_cart reported value: 0 on a visibly full trolley**
>
> Client storage is read through `useSyncExternalStore`, which deliberately renders the *server* snapshot during hydration so the markup matches, then re-renders with the client snapshot. Effects run against that first, server-shaped render — so the one-shot guard fired against an *empty* trolley, marked itself done, and blocked the correct value that arrived a moment later.
>
> The page looked perfect. The event was empty. Fixed with an explicit `useStoreHydrated()` gate on every storage-dependent decision.

> **2. Redirect guards bounced signed-in shoppers to /login**
>
> The same root cause with a user-visible symptom: the checkout and slots pages redirect when there is no user, and during the hydration render the user is always the server snapshot — `null`. A signed-in shopper reaching checkout was sent back to sign in.
>
> A green build proves the code compiles. It proves nothing about whether a tag fired, what it carried, or whether the journey works.

Duplicate-event checks

- Walk the full funnel and confirm exactly one of each event per step.
- Refresh the confirmation page repeatedly — `purchase` must fire once and never again.
- Work a stepper 0 → 1 → 2 → 3 → 2 and confirm four events, each with `quantity: 1`, never the running total.
- Change the aisle sort and confirm `view_item_list` re-fires with updated `index` values.

15. GTM container configuration

Everything needed to rebuild container `GTM-5PDKGPN9` from this document alone.

Variables

Name	Type	Value / key
Const - GA4 Measurement ID	Constant	G-JSJ85T9LV1
DLV - ecommerce	Data Layer Variable	ecommerce
DLV - user_id	Data Layer Variable	user_id
DLV - user_properties	Data Layer Variable	user_properties
DLV - slot_day	Data Layer Variable	slot_day
DLV - slot_window	Data Layer Variable	slot_window
DLV - slot_fee	Data Layer Variable	slot_fee
DLV - search_results_count	Data Layer Variable	search_results_count
DLV - consent_action	Data Layer Variable	consent_action
Event	Built-in	{{Event}}

Triggers

Name	Type	Condition
Initialization - All Pages	Initialization	—
Consent Initialization - All Pages	Consent Initialization	—
CE - All Freshcart Events	Custom Event (regex)	^(view_item_list|select_item|view_item|add_to_cart|remove_from_cart|view_cart|begin_checkout|add_shipping_info|add_payment_info|purchase|view_promotic
CE - purchase	Custom Event	purchase
All Pages	Page View	—

Tags

Tag	Type	Trigger	Configuration
Google tag	Google tag	Initialization - All Pages	Tag ID `G-JSJ85T9LV1` . `send_page_view` left at its default **true**. `user_id` set as a configuration setting from `DLV - user_id` — *not* inside user properties.
GA4 Event - Freshcart	GA4 Event	CE - All Freshcart Events	Event name `{{Event}}` . Send ecommerce data from the Data Layer. User properties mapped from `DLV - user_properties` .
Consent Initialization	Custom HTML / CMP template	Consent Initialization - All Pages	Defaults denied. Must run before every other tag.
Google Ads - Conversion	Google Ads Conversion Tracking	CE - purchase	Order ID from `transaction_id` for de-duplication. Value and currency from the ecommerce object.
Google Ads - Remarketing	Google Ads Remarketing	All Pages	Conversion ID only — never the conversion label. No URL passthrough support.
Conversion Linker	Conversion Linker	All Pages	Prerequisite for both Ads tags. Not optional.

> **No page_view tag in this container**
> Do not create a GA4 Event tag for `page_view` , and do not set `send_page_view: false` on the Google tag. Freshcart is a multi-page app; the automatic page view is correct and sufficient (§2).

Every tag is configured with GTM's built-in consent checks, so nothing fires under denied `analytics_storage` or `ad_storage` .

16. Google Ads

Phase 1 is web-side only. Two tags: a conversion on `purchase` , keyed on `transaction_id` so a repeated confirmation cannot double-count; and a remarketing tag on all pages.

A **Conversion Linker** tag is a documented hard prerequisite for both — without it, conversions silently fail to attribute and nothing surfaces an error.

Do not put the conversion label on the remarketing tag; Google is explicit that a remarketing tag takes the conversion ID alone.

Status: the conversion ID and label were not yet issued at the time of writing. The plumbing is wired via `NEXT_PUBLIC_FRESHCART_ADS_CONVERSION_ID` and `NEXT_PUBLIC_FRESHCART_ADS_CONVERSION_LABEL` ; supplying them requires no code change.

17. Migrating a legacy data layer into items[]

> **Freshcart does not need this section, and that is the point**
> Everything above specifies a data layer designed alongside the code that reads it. Its events arrive GA4-shaped, so no translation is required. This section and the next describe the opposite situation: an estate that already exists, whose shape you did not choose and cannot change on your own timetable. That is the more common brief, and the container is usually the only place you are permitted to fix it.

Tag-management estates that predate GA4 typically describe products as **index-aligned parallel arrays** on a flat object: one array per attribute, one element per product, joined by position. GA4 wants the transpose — an array of objects, each a complete product. The join has to happen somewhere.

When the page cannot be changed, it happens in the container.

THE INHERITED OBJECT. FICTIONAL, BUT THE SHAPE IS TYPICAL.

```
window.udo = {
  page_type:      "basket",
  page_channel:   "Groceries",
  basket_currency: "GBP",
  basket_value:   "8.85",

  // One array per attribute. Element 1 of each describes the same product.
  line_ref:       ["8412", "9930", "1177"],
  line_desc:      ["Semi-skimmed milk, 2L", "Sourdough bloomer, 800g", "Free-range eggs, 12"],
  line_dept:      ["Dairy & eggs", "Bakery", "Dairy & eggs"],
  line_aisle:     ["Milk", "Loaves", "Eggs"],
  line_unit_net:  ["1.45", "2.20", "3.75"],
  line_qty:       ["2", "1", "1"],
  line_offer:     ["", "BLOOMER2FOR3", ""]
};
```

Three properties of it drive everything that follows.

- **Everything is a string**, including money and counts. GA4 wants `price` and `quantity` as numbers.
- **Position is the only join.** Nothing structural connects `line_ref[1]` to `line_desc[1]`. They describe the same product by convention, and a convention cannot be validated.
- **Present-but-empty is not absent.** `line_offer[0]` is an empty string, meaning “this line carries no offer”. A missing `line_offer` key would mean the estate never emits offers at all. Two different facts. §18 is about what it costs to conflate them.

Two questions, two different tests

The variable tests presence twice, at two levels, and the tests are deliberately not the same.

- **Schema level** — `'line_offer' in udo`. Does this estate emit the field at all? An empty array is falsy but is a perfectly good answer meaning “no lines on this page”. A truthiness test would report it identically to a field the estate has never populated, hiding a genuine specification gap forever.
- **Row level** — `raw === ''`. Does *this line* carry a value? An empty string here is true and meaningful, but GA4 has no use for it, so the key is omitted rather than sent as noise.

The transformation

A GTM **Custom JavaScript Variable**, returned into the GA4 event tag's `items` field. It runs in page context, so it can read `window` directly and use ordinary constructors.

CUSTOM JAVASCRIPT VARIABLE — UDO.ITEMS

```
function () {
  var udo = window.udo;
  if (!udo) return undefined;

  // Legacy key -> GA4 item key -> coercion. The join is the index, nothing else.
  var MAP = [
    ['line_ref',      'item_id',        'string'],
    ['line_desc',     'item_name',      'string'],
    ['line_dept',     'item_category',  'string'],
    ['line_aisle',    'item_category2', 'string'],
    ['line_unit_net', 'price',          'number'],
    ['line_qty',      'quantity',       'number'],
    ['line_offer',    'coupon',         'string']
  ];

  // SCHEMA LEVEL: does the estate emit this key at all? `in`, never truthiness.
  var cols = [];
  for (var m = 0; m < MAP.length; m++) {
    var src = MAP[m][0];
    if (!(src in udo)) continue;
    if (Object.prototype.toString.call(udo[src]) !== '[object Array]') continue;
    cols.push({ out: MAP[m][1], as: MAP[m][2], values: udo[src] });
  }

  // GA4 requires item_id or item_name on every element, so item_id anchors the set.
  var anchor = null;
  for (var c = 0; c < cols.length; c++) {
    if (cols[c].out === 'item_id') anchor = cols[c];
  }
  if (!anchor) return undefined;

  // ALIGNMENT: unequal lengths are a defect, not a shape to accommodate.
  var rows = anchor.values.length;
  for (var k = 0; k < cols.length; k++) {
    if (cols[k].values.length !== rows) return undefined;   // fail closed
  }

  var items = [];
  for (var i = 0; i < rows; i++) {
    var item = {};
    for (var j = 0; j < cols.length; j++) {
      var raw = cols[j].values[i];

      // ROW LEVEL: '' is a true answer, but GA4 has no use for it. Omit the key.
      if (raw === undefined || raw === null || raw === '') continue;

      if (cols[j].as === 'number') {
        var n = Number(raw);            // never parseFloat: '1.45kg' would pass
        if (isNaN(n)) continue;
        item[cols[j].out] = n;
      } else {
        item[cols[j].out] = String(raw);
      }
    }
    items.push(item);
  }

  return items.length ? items : undefined;
}
```

ITS ACTUAL RETURN VALUE AGAINST THE OBJECT ABOVE


```
[
  {
    "item_id": "8412",
    "item_name": "Semi-skimmed milk, 2L",
    "item_category": "Dairy & eggs",
    "item_category2": "Milk",
    "price": 1.45,
    "quantity": 2
  },
  {
    "item_id": "9930",
    "item_name": "Sourdough bloomer, 800g",
    "item_category": "Bakery",
    "item_category2": "Loaves",
    "price": 2.2,
    "quantity": 1,
    "coupon": "BLOOMER2FOR3"
  },
  {
    "item_id": "1177",
    "item_name": "Free-range eggs, 12",
    "item_category": "Dairy & eggs",
    "item_category2": "Eggs",
    "price": 3.75,
    "quantity": 1
  }
]
```


Note `price: 2.2` on the second line. The estate held `"2.20"`; the coercion is real, and the trailing zero was never a number in the first place. Note also that `coupon` appears on the second line only — the two empty strings were dropped rather than sent as empty values.

What each trap costs

None of these raises an error. Every one produces a payload that looks reasonable in Tag Assistant.

Trap	What happens	What to do instead
Length mismatch between arrays	One array is shorter, so every product after that point joins to the wrong attributes.	Compare lengths against the anchor and return nothing. Misaligned data reports a plausible total and is never questioned; missing data gets chased.
`undefined` at an index	Ambiguous — it means either 'no such product' or 'this array is shorter than the others'. Reading one array cannot tell you which.	Resolve it at the set level with the alignment check, before reading any row.
Truthiness used to test presence	`""`, `0`, `"0"` and `[]` all mislead: two are falsy but meaningful, two are truthy but not what you meant.	`in` for schema questions, an explicit value comparison for row questions. See §18.
Strings left uncoerced	`price: "1.45"` is accepted by the collection endpoint. GA4 then treats revenue as text, and sums come back empty rather than wrong.	Coerce at the boundary with `Number()`, and drop `NaN` rather than sending it.
`parseFloat` instead of `Number`	`parseFloat("1.45kg")` returns `1.45`, so a unit accidentally left on the price passes silently for years.	`Number("1.45kg")` is `NaN`. Prefer the coercion that fails.
Array index reused as GA4 `index`	GA4's `index` means position in a displayed list. The array index is a join key. They coincide on a list page and diverge everywhere else.	Let the event that has a list set `index`. The transformation should not invent one.

> **Fail closed, not quietly wrong**
> When the arrays do not align, the variable returns `undefined` and the tag sends no items. That is deliberate. A tag sending no items is a visible gap that somebody chases. A tag sending misaligned items attributes the milk's price to the eggs, reports a plausible total, and is never questioned.

Limits worth knowing before you design the mapping

- The `items` array accepts at most **200 elements**. A large grocery basket can reach that; decide deliberately what to drop rather than discovering the truncation later.
- At most **27 custom item parameters** beyond the prescribed ones, of which only **10** can become item-scoped custom dimensions on a standard property (25 on 360). A legacy object with forty product fields will not map one-to-one, so the mapping is a prioritisation exercise, not a transcription.
- Every element needs `item_id` or `item_name`. That is why the variable anchors on `item_id` and returns nothing without it.

If this moves into a custom template

The code above is a Custom JavaScript Variable, which runs in page context. A **custom template** is a different environment and the code does not port unchanged: sandboxed JavaScript is a subset of ECMAScript 5.1 with no `window`, no `new`, and no global constructors — `Number()` and `String()` are simply not available. Reading the object requires `copyFromWindow`, which returns a deep copy and is expensive on an object this size, and the coercions come from the sandbox's own standard library instead.

This is the same distinction that governs consent: page code uses a `gtag` command, a custom template must use Tag Manager's `updateConsentState` API. Knowing which environment you are writing for is half of getting either right.

18. Absence is not emptiness

The transformation in §17 tests presence with `in` rather than truthiness. That choice is worth its own section, because the same distinction governs *trigger conditions*, and there it is responsible for a failure mode that is both expensive and invisible.

The operators do not agree, and none of them is wrong

Four ways to ask “is there a value here?”, over every input that behaves surprisingly. Each cell is the result of evaluating the expression.

Value of o.k	'k' in o	!!o.k	o.k === undefined	o.k != null
key not present	false	false	true	false
undefined	true	false	true	false
null	true	false	false	false
""	true	false	false	true
0	true	false	false	true
"0"	true	true	false	true
false	true	false	false	true
[]	true	true	false	true
"false"	true	true	false	true

The second row is the interesting one. A key explicitly set to `undefined` is **present** by `in` and **missing** by `=== undefined`. Both answers are correct; they are answers to different questions. `in` asks whether the estate declared the field. `=== undefined` asks whether it holds anything.

The rows that catch people in practice are `"0"` and `[]`, which are truthy while meaning “zero” and “nothing”, and `0` and `""`, which are falsy while being real values the estate deliberately sent.

Negation is not the mirror of equality

A GTM condition reads a variable. When the underlying field is absent, that variable resolves to `undefined` — and `undefined` is not equal to anything, so it **satisfies the negation**. Split an estate into three groups and the asymmetry is immediate.

Condition	Fires on	Why
equals "Groceries"	**P**	Only where the field exists and matches. Absence cannot satisfy it.
does not equal "Groceries"	**Q + R**	Fires on every page where the field is missing, because an absent variable is not equal to anything.
contains "Grocer"	**subset of P + Q**	Same shape as `equals` — a value must exist to contain anything.
does not contain "Grocer"	**Q + R**	Same trap as `does not equal`, and easier to miss because it reads like a filter rather than a negation.

Where **P** is the set of pages on which the field is present and matches, **Q** those where it is present and differs, and **R** those where it does not exist at all. `equals` fires on P. Its negation fires on Q *plus R* — not the complement you drew on the whiteboard, but the complement plus the entire remainder of the estate.

> **This is documented behaviour, not a quirk — Google relies on it**
>
> Google's own privacy guidance builds an Analytics opt-out on exactly this. The recipe sets a cookie only for users who *have* opted out, then fires the tag on the condition `google-analytics-opt-out cookie does not equal true`. For everyone else the cookie is absent, and the tag fires *because* an absent value does not equal `true`.
>
> So the rule is not "never negate". Negation over an absent field is a legitimate and documented technique. The rule is that you must know what absence means on your estate before you rely on it.

Why it is expensive, and why nobody notices

Take a field that exists only on product pages — a small fraction of a large estate. A tag conditioned on `does not equal` against that field fires on almost every page view on the domain: all the pages where the value genuinely differs, plus every page that never had the field.

Three things then combine to keep it hidden.

- **The tag is firing**, which looks like success. Nothing errors, and Tag Assistant shows the tag as fired because it genuinely did.
- **The condition is valid.** The interface has nothing to warn about — the operator exists, the variable exists, the syntax is correct.
- **The cost lands somewhere else.** If the tag is a third-party pixel billed per event, or a server-side hit, the bill arrives on a different team's budget than the one that configured it.

This is the same class of failure as the one recorded against this project in §14: a push that reaches no tag and a tag that fires on the wrong pages are both invisible in the browser, because the dataLayer shows what the page said, never what the container did about it.

> **Rule: build on presence or on equality, never on a negation over a field you have not proved is populated everywhere**
>
> If both directions are genuinely needed, key them on a field that is present on *every* page in the container — a page type or channel field — rather than on the sparse one. Freshcart's own triggers key on `{{Event}}`, which is never absent, which is why none of this bites here.
>
> "Proved is populated everywhere" means measured across a representative crawl. It does not mean the specification says the field is mandatory. The specification is what the estate was supposed to do.

Proving it: a presence audit

Run this on every page of a representative crawl before writing a condition that depends on a field being there. It records presence and never values, so it is safe on pages carrying personal data.

PRESENCE AUDIT — RECORDS ABSENT / EMPTY STRING / NULL / EMPTY ARRAY / PRESENT

```
/* Records PRESENCE, never the value — so it is safe to run on pages
   carrying personal data. Paste on each page of a representative crawl. */
(function (fields) {
  var dl = window.dataLayer || [];

  /* A flat merge is a SIMPLIFICATION of GTM's model, not a replica of it:
     GTM tracks values per event, and `ecommerce: null` resets rather than
     merges. It is accurate enough to answer "does this key ever appear",
     which is the only question being asked here. */
  var merged = {};
  for (var i = 0; i < dl.length; i++) {
    if (Object.prototype.toString.call(dl[i]) !== '[object Object]') continue;
    for (var k in dl[i]) merged[k] = dl[i][k];
  }

  var row = { path: location.pathname };
  for (var f = 0; f < fields.length; f++) {
    var name = fields[f];
    row[name] =
      !(name in merged)        ? 'ABSENT'        :
      merged[name] === ''      ? 'EMPTY STRING'  :
      merged[name] === null    ? 'NULL'          :
      Array.isArray(merged[name]) && !merged[name].length ? 'EMPTY ARRAY' :
      'present';
  }
  console.table([row]);
  return row;
})(['page_type', 'page_channel', 'basket_currency']);
```

Read the output as three distinct findings, not one. **ABSENT** on some pages means any negated condition over that field will fire there. **EMPTY STRING** means the estate is emitting the field and has nothing to put in it, which is a content problem rather than a tagging one. **ABSENT everywhere** means the field was specified and never implemented — worth knowing before you build a report on it.

19. Appendix A: complete event list

All 21 event names the code pushes. Generated from `FRESHCART_EVENT_NAMES` in the implementation, so it cannot fall out of step. Note the absence of `page_view` and `view_search_results` — see §11.

view_item_list	select_item	view_item
add_to_cart	remove_from_cart	view_cart
begin_checkout	add_shipping_info	add_payment_info
purchase	view_promotion	select_promotion
login	sign_up	search
select_content	slot_selected	substitution_preference_changed
minimum_spend_not_met	consent_choice	consent_update