DEVELOPER DOCUMENTATION

Event Counter API

Shopware extension **SyseaEventCounter** 1.0.1 · Shopware 6.6.10 and newer, Shopware 6.7 · Updated 2026-10-12

Event Counter counts what happens in a Shopware shop as anonymous daily totals, for example product page views, wishlist additions, checkout steps and orders. This document describes how to read those totals from other systems, how to manage events through the Admin API, and how theme and plugin developers can count their own storefront events.

Contents

1. Overview
2. Authentication
3. Conventions
4. Report (JSON / CSV)
5. Statistics
6. Catalog
7. Entities
8. Managing events
9. Trigger reference
10. Storefront integration
11. Recipes
12. Errors
13. Data, retention, privacy
14. Changelog

1. Overview

The extension stores **one row per UTC day, sales channel and event** with a counter. There are no visitor IDs, sessions, IP addresses or other personal data, only totals such as "Product page viewed: 918 on 2026-10-01 in sales channel Storefront".

Every event has a stable **technical name** (`order_placed`, `wishlist_add`, ...) and one of two scopes:

- **browser**: counted by the storefront script after the visitor has accepted "Anonymous usage statistics" in the cookie banner (clicks, page views, storefront JavaScript events).
- **server**: counted by the shop itself from requests the visitor sends anyway (Flow Builder events such as "order placed", uncached storefront routes, adding to the cart and the wishlist). No consent is involved.

The extension ships 27 ready-made events (inactive until the merchant switches them on) and lets merchants define their own.

Report
Flat rows (day, sales channel, event, count) as JSON or CSV for BI tools and spreadsheets.

Statistics

Server-side aggregations: totals with the previous period, time series, per event, per sales channel.

Entities

Standard Admin API entities for search, filters, aggregations and for creating or changing events.

Storefront integration

Count your own theme or plugin events with a DOM event, without writing to the API.

All endpoints belong to the Shopware **Admin API** (`/api/…`) and use Shopware's access control. There is no Store API endpoint; the counts are not exposed to the storefront.

---

2. Authentication

Use a Shopware **integration** with a role that may only read the event counter. This is the recommended setup for Looker Studio, Power BI, scripts and other external tools.

1. **Create a role.** Settings › System › Users & permissions › Roles › *Create role*. Under *Marketing › Event counter* tick **View**. Nothing else is needed.
2. **Create an integration.** Settings › System › Integrations › *Add integration*. Leave *Administrator* switched off, assign the role from step 1 and copy the *Access key ID* and the *Secret access key*.
3. **Request a token** with the client credentials grant and send it as a bearer token.

`POST` `/api/oauth/token`

```
curl -X POST "https://shop.example.com/api/oauth/token" \
  -H "Content-Type: application/json" \
  -d '{"grant_type":"client_credentials","client_id":"SWIA…","client_secret":"…"}'
```

```
{
  "token_type": "Bearer",
  "expires_in": 600,
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGci…"
}
```

The token is valid for **600 seconds**. Request a new one when a call answers `401`.

```
curl "https://shop.example.com/api/_action/sysea-event-counter/report" \
  -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGci…" \
  -H "Accept: application/json"
```

Privileges

The administration offers four permissions under *Marketing › Event counter*. They map to these entity privileges:

Permission	Privileges	Allows
View	`sysea_event_counter_daily:read`, `sysea_event_counter_definition:read`, `sysea_event_counter_definition_translation:read`, `sysea_event_counter_definition_sales_channel:read`, `sales_channel:read`, `language:read`, and `user_config:read` / `create` / `update` for saved list filters and columns, as in Shopware's own modules	All read endpoints in this document
Edit	View plus `sysea_event_counter_definition:update`, create/update of translations and sales channel assignments	Change events, switch them on and off
Create	Edit plus `sysea_event_counter_definition:create`	Create own events
Delete	View plus `sysea_event_counter_definition:delete`	Delete events (their counts are kept)

A server event with a Flow Builder trigger is counted through a flow that is saved along with the event. The event counter permissions deliberately contain no flow privileges, because those would allow changing every flow of the shop. Creating or changing such an event therefore also needs the permission *Settings › Flow Builder › Create* (`flow:create`, `flow:update`, `flow_sequence:create`, `flow_sequence:update`); deleting it needs *Flow Builder › Delete* (`flow:delete`). Without them the call answers `403` and nothing is written; in the administration such events are read-only and the Flow Builder trigger is not offered. An integration with only *View* cannot read anything else in the shop. A request for orders, for example, answers `403` with `"missingPrivileges":["order:read"]`.

3. Conventions

Topic	Rule
Time zone	Days are **UTC calendar days**. Every response states `"timezone": "UTC"`.
Dates	`YYYY-MM-DD`. `from` and `to` are both inclusive.
Default period	`to` = today (UTC), `from` = `to` minus 29 days, i.e. the last 30 days including today.
Invalid dates	A missing or malformed date falls back to its default. No error is raised, so check the `from` / `to` echoed in the response.
Maximum range	3,660 days. A longer range, or `from` after `to`, returns `"data": []` with status 200.
List parameters	Repeat the key with brackets (`technicalName[]=cart_add&technicalName[]=order_placed`) or pass a comma-separated value (`technicalName=cart_add,order_placed`). URL-encode the brackets where your client requires it (`%5B%5D`).
IDs	Sales channel IDs are 32-character lowercase hex strings, as everywhere in Shopware.
Labels	Event and sales channel names follow the `sw-language-id` request header. Without it, or without a translation, the system language is used, then the technical name.
Empty days	Days without a count have **no row** and no bucket. Fill gaps on your side when you need a continuous series.
Today	Today's numbers are live and still grow during the day.

4. Report (JSON / CSV)

`GET` `/api/_action/sysea-event-counter/report` needs `sysea_event_counter_daily:read`

Flat rows in **long format**: one row per day, sales channel and event. New events never change the column layout, and BI tools pivot the data themselves. This is the endpoint for exports and imports.

Parameter	Type	Description
`from`	date	First day, inclusive. Default: 29 days before `to`.
`to`	date	Last day, inclusive. Default: today (UTC).
`salesChannelId[]`	list of IDs	Only these sales channels. Default: all.
`technicalName[]`	list of names	Only these events. Default: all.
`format`	`json` | `csv`	Default: `json`.

JSON

```
GET /api/_action/sysea-event-counter/report?from=2026-10-01&to=2026-10-
02&technicalName[]=order_placed&technicalName[]=product_view
```


```
{
  "from": "2026-10-01",
  "to": "2026-10-02",
  "timezone": "UTC",
  "data": [
    {
      "event_date": "2026-10-01",
      "sales_channel_id": "019f313627b3724d85de11e1e6a385a1",
      "sales_channel_name": "Storefront",
      "event_name": "order_placed",
      "event_label": "Order placed",
      "event_count": 28
    },
    {
      "event_date": "2026-10-01",
      "sales_channel_id": "019f313627b3724d85de11e1e6a385a1",
      "sales_channel_name": "Storefront",
      "event_name": "product_view",
      "event_label": "Product page viewed",
      "event_count": 918
    },
    { "event_date": "2026-10-02", …, "event_name": "order_placed", "event_count": 23 },
    { "event_date": "2026-10-02", …, "event_name": "product_view", "event_count": 697 }
  ]
}
```


Field	Type	Description
`event_date`	string	UTC day, `YYYY-MM-DD`
`sales_channel_id`	string	Sales channel ID (hex)
`sales_channel_name`	string	Sales channel name in the request language
`event_name`	string	Technical name of the event
`event_label`	string	Event name in the request language; the technical name if the event was deleted
`event_count`	integer	Count for that day, sales channel and event

Rows are sorted by `event_date`, then sales channel name, then technical name.

CSV

```
GET /api/_action/sysea-event-counter/report?from=2026-10-01&to=2026-10-
02&format=csv&technicalName[]=cart_add&technicalName[]=order_placed
```

```
event_date,sales_channel_id,sales_channel_name,event_name,event_label,event_count
2026-10-01,019f313627b3724d85de11e1e6a385a1,Storefront,cart_add,"Product added to cart",89
2026-10-01,019f313627b3724d85de11e1e6a385a1,Storefront,order_placed,"Order placed",28
2026-10-02,019f313627b3724d85de11e1e6a385a1,Storefront,cart_add,"Product added to cart",66
2026-10-02,019f313627b3724d85de11e1e6a385a1,Storefront,order_placed,"Order placed",23
```

- RFC 4180: comma separator, header row, CRLF line endings, fields quoted where needed.
- UTF-8 without byte order mark. Line breaks inside values are replaced by spaces, because Looker Studio rejects multi-line fields.
- Text that starts with `=`, `+`, `-`, `@`, a tab or a carriage return gets a leading apostrophe (`'=…`), so spreadsheets show it as text instead of running it as a formula (CSV injection). Counts are not affected.
- Headers: `Content-Type: text/csv; charset=utf-8` and `Content-Disposition: attachment; filename="event-counter_2026-10-01_2026-10-02.csv"`.
- The columns are the same as in JSON. With `sw-language-id` set to German, `event_label` reads e.g. `"Bestellung abgeschlossen"`.

Excel with a German locale expects semicolons when a CSV file is opened by double-click. Use *Data › From Text/CSV* instead; it detects the comma and UTF-8 correctly.

---

5. Statistics

`GET` `/api/_action/sysea-event-counter/statistics` needs `sysea_event_counter_daily:read`

Aggregations computed in the database. The administration's statistics pages use this endpoint, and you can use it for dashboards without downloading every daily row.

Parameter	Type	Description
`dimension`	enum	`summary`, `time`, `event`, `event_time`, `sales_channel`. Default and fallback for unknown values: `time`.
`from`, `to`	date	As in the report.
`groupBy`	enum	`day` (default), `week`, `month`. Used by `time` and `event_time`.
`scope`	enum	`browser` or `server`. Omit for both.
`salesChannelId[]`	list of IDs	Only these sales channels.
`technicalName[]`	list of names	Only these events.

Every response has the same envelope: `from`, `to`, `timezone`, `groupBy` (as applied) and `data`.

- **Buckets:** `day` buckets are the date itself. `week` buckets are the Monday of the ISO week. `month` buckets are the first day of the month (`2026-10-01`). A partial first or last week or month only contains the days inside `from` … `to`.
- **Scope filter:** `scope` needs the event definition. Counts of deleted events have no scope and are left out when `scope` is set.

dimension=summary

Key figures for the period, compared with the period of the same length directly before it.

```
GET /api/_action/sysea-event-counter/statistics?dimension=summary&from=2026-10-01&to=2026-10-07
```


```
{
  "from": "2026-10-01",
  "to": "2026-10-07",
  "timezone": "UTC",
  "groupBy": "day",
  "data": {
    "total": 18985,
    "browser": 15795,
    "server": 3190,
    "events": 12,
    "days": 7,
    "averagePerDay": 2712.1,
    "previousTotal": 19489,
    "previousFrom": "2026-09-24",
    "previousTo": "2026-09-30",
    "topEvent": {
      "technicalName": "product_view",
      "label": "Product page viewed",
      "scope": "browser",
      "count": 6314
    }
  }
}
```


Field	Description
`total`, `browser`, `server`	Sum of all counts, and split by scope
`events`	Number of distinct events with at least one count
`days`, `averagePerDay`	Days in the period and `total / days`, rounded to one decimal
`previousTotal`, `previousFrom`, `previousTo`	The same figure for the preceding period
`topEvent`	Event with the highest count, or `null`

dimension=time

One series: the sum of the selected events per bucket.

```
GET /api/_action/sysea-event-counter/statistics?dimension=time&from=2026-10-01&to=2026-10-07&technicalName[]=order_placed
```


```
{
  "from": "2026-10-01", "to": "2026-10-07", "timezone": "UTC", "groupBy": "day",
  "data": [
    { "bucket": "2026-10-01", "count": 28 },
    { "bucket": "2026-10-02", "count": 23 },
    { "bucket": "2026-10-03", "count": 39 },
    { "bucket": "2026-10-04", "count": 39 },
    { "bucket": "2026-10-05", "count": 32 },
    { "bucket": "2026-10-06", "count": 35 },
    { "bucket": "2026-10-07", "count": 30 }
  ]
}
```


dimension=event

Total per event, highest first. Useful for rankings and funnels (pass the funnel steps as `technicalName[]`).

```
GET /api/_action/sysea-event-counter/statistics?dimension=event&from=2026-10-01&to=2026-10-07&scope=server
```


```
{
  "from": "2026-10-01", "to": "2026-10-07", "timezone": "UTC", "groupBy": "day",
  "data": [
    { "technicalName": "cart_view", "label": "Cart viewed", "scope": "server", "count": 987 },
    { "technicalName": "search_view", "label": "Search results viewed", "scope": "server", "count": 896 },
    { "technicalName": "cart_add", "label": "Product added to cart", "scope": "server", "count": 672 },
    { "technicalName": "checkout_confirm_view", "label": "Checkout: confirmation viewed", "scope": "server", "count": 382
},
    { "technicalName": "order_placed", "label": "Order placed", "scope": "server", "count": 226 },
    { "technicalName": "newsletter_register", "label": "Newsletter sign-up", "scope": "server", "count": 27 }
  ]
}
```


`scope` is `null` for counts of events that have since been deleted.

dimension=event_time

One series per event, ordered by total, highest first (the same order as `dimension=event`). Without `technicalName[]`, the eight events with the highest totals are returned.

```
GET /api/_action/sysea-event-counter/statistics?dimension=event_time&groupBy=week&from=2026-09-07&to=2026-10-
04&technicalName[]=cart_add&technicalName[]=order_placed
```

```
{
  "from": "2026-09-07", "to": "2026-10-04", "timezone": "UTC", "groupBy": "week",
  "data": [
    {
      "technicalName": "cart_add",
      "label": "Product added to cart",
      "data": [
        { "bucket": "2026-09-07", "count": 622 },
        { "bucket": "2026-09-14", "count": 607 },
        { "bucket": "2026-09-21", "count": 697 },
        { "bucket": "2026-09-28", "count": 663 }
      ]
    },
    {
      "technicalName": "order_placed",
      "label": "Order placed",
      "data": [
        { "bucket": "2026-09-07", "count": 226 },
        { "bucket": "2026-09-14", "count": 212 },
        { "bucket": "2026-09-21", "count": 241 },
        { "bucket": "2026-09-28", "count": 217 }
      ]
    }
  ]
}
```


dimension=sales_channel

Total per sales channel, highest first.

```
GET /api/_action/sysea-event-counter/statistics?dimension=sales_channel&from=2026-10-01&to=2026-10-07
```


```
{
  "from": "2026-10-01", "to": "2026-10-07", "timezone": "UTC", "groupBy": "day",
  "data": [
    { "salesChannelId": "019f313627b3724d85de11e1e6a385a1", "name": "Storefront", "count": 18985 }
  ]
}
```


6. Catalog

`GET` `/api/_action/sysea-event-counter/catalog` needs `sysea_event_counter_definition:read`

Lists what the installed Shopware version offers as triggers. The administration uses it for the selection lists in the trigger dialog; use it to build valid triggers for your own events.

```
{
  "shopwareVersion": "6.7.11.1",
  "catalogVersion": "6.7",
  "jsEvents": [
    { "eventName": "beforeFormSubmit", "selector": "[data-add-to-cart]", "plugin": "AddToCart" },
    { "eventName": "Wishlist/onProductAdded", "selector": "[data-wishlist-storage]", "plugin": "WishlistStorage" },
    …
  ],
  "routes": [
    { "name": "frontend.account.address.create", "path": "/account/address/create", "methods": ["POST"], "httpCache":
false, "xhr": false },
    …
  ],
  "businessEvents": [
    { "name": "checkout.customer.before.login" },
    { "name": "checkout.customer.deleted" },
    …
  ],
  "coreEvents": ["cart_add", "wishlist_add"]
}
```


Key	Contents	Use for
`jsEvents`	Events of Shopware's storefront JavaScript plugins with the element they are fired on. On 6.7.11: 105 entries.	`dom_event` triggers
`routes`	Storefront routes of this shop, live from the router, including routes of other extensions. `httpCache: true` means the page is usually served from the HTTP cache.	`route` and `page_view` triggers
`businessEvents`	Flow Builder events of this shop, including `state_enter.*` and `state_leave.*` events.	`business_event` triggers
`coreEvents`	Core events the extension listens to directly.	`core_event` triggers

`jsEvents` come from a list the extension ships per Shopware minor version (`catalogVersion`); routes and business events are read live.

7. Entities

Both entities are regular Shopware entities, available through the standard Admin API endpoints: `/api/search/…`, `/api/search-ids/…` and, for definitions, `POST` / `PATCH` / `DELETE`. Filters, sorting, `includes` and aggregations work as described in the Shopware Admin API reference.

sysea_event_counter_daily

`POST` `/api/search/sysea-event-counter-daily` needs `sysea_event_counter_daily:read`

Field	Type	Description
`id`	ID	Row ID
`eventDate`	date	UTC day. Serialized as `2026-09-30T00:00:00.000+00:00`.
`salesChannelId`	ID	Sales channel
`technicalName`	string	Technical name of the event. No foreign key: the history stays when an event is deleted.
`count`	integer	Count of that day
`salesChannel`	association	The sales channel entity
`createdAt`, `updatedAt`	datetime	Row created / last increased

Example: orders in September, newest day first, with the monthly sum as an aggregation.

```
POST /api/search/sysea-event-counter-daily
{
  "limit": 2,
  "filter": [
    { "type": "equals", "field": "technicalName", "value": "order_placed" },
    { "type": "range", "field": "eventDate", "parameters": { "gte": "2026-09-01", "lte": "2026-09-30" } }
  ],
  "sort": [{ "field": "eventDate", "order": "DESC" }],
  "aggregations": [{ "name": "orders", "type": "sum", "field": "count" }],
  "includes": { "sysea_event_counter_daily": ["eventDate", "salesChannelId", "technicalName", "count"] }
}
```

```
{
  "total": 2,
  "data": [
    { "eventDate": "2026-09-30T00:00:00.000+00:00", "salesChannelId": "019f313627b3724d85de11e1e6a385a1", "technicalName":
"order_placed", "count": 26, "apiAlias": "sysea_event_counter_daily" },
    { "eventDate": "2026-09-29T00:00:00.000+00:00", "salesChannelId": "019f313627b3724d85de11e1e6a385a1", "technicalName":
"order_placed", "count": 31, "apiAlias": "sysea_event_counter_daily" }
  ],
  "aggregations": {
    "orders": { "name": "orders", "sum": 936, "apiAlias": "orders_aggregation" }
  }
}
```

Treat this entity as **read-only**. Counts are written by the extension, which applies the bot filter, the consent rules and the per-request limits. Do not grant create, update or delete on it to integrations.

sysea_event_counter_definition

`POST` `/api/search/sysea-event-counter-definition` needs `sysea_event_counter_definition:read`

Field	Type	Description
`id`	ID	Event ID
`technicalName`	string, required	`^[a-z][a-z0-9_]{2,63}$`, unique, **cannot be changed** after creation (it identifies the history)
`name`	translated string, required	Display name, e.g. "Order placed"
`active`	bool	Only active events count. Default `false`.
`scope`	`browser` | `server`, required	Where the event is counted; determines the allowed trigger types
`triggers`	list of objects	When the event counts, see Trigger reference. Up to 20.
`oncePerPage`	bool	Browser events: count at most once per page view. Default `true`.
`presetKey`	string | null	Set on the 27 ready-made events, `null` on your own
`salesChannels`	many-to-many	Restrict to these sales channels. **Empty = all sales channels.**
`translations`	association	Names per language


```
{
  "id": "7d3e2674c6f1f777b17131c0559cfb91",
  "technicalName": "wishlist_add",
  "name": "Product added to wishlist (incl. guests)",
  "active": true,
  "scope": "browser",
  "triggers": [
    { "type": "dom_event", "eventName": "Wishlist/onProductAdded", "selector": "[data-wishlist-storage]" }
  ],
  "oncePerPage": false,
  "presetKey": "wishlist_add",
  "salesChannels": [],
  "apiAlias": "sysea_event_counter_definition"
}
```


8. Managing events

Create, change and delete events with the standard entity endpoints. The extension validates every write and keeps its storefront configuration and managed flows in sync.

Create a browser event

`POST` `/api/sysea-event-counter-definition` needs the *Create* permission

```
POST /api/sysea-event-counter-definition?_response=basic
{
  "technicalName": "newsletter_popup_open",
  "name": "Newsletter popup opened",
  "active": true,
  "scope": "browser",
  "oncePerPage": true,
  "triggers": [
    { "type": "dom_event", "eventName": "shown.bs.modal", "selector": ".newsletter-popup" }
  ]
}
```

Without `_response` the call answers `204 No Content`; with it, the created entity is returned.

Create a server event from a Flow Builder event

```
POST /api/sysea-event-counter-definition
{
  "technicalName": "order_cancelled",
  "name": "Order cancelled",
  "active": true,
  "scope": "server",
  "triggers": [
    { "type": "business_event", "eventName": "state_enter.order.state.cancelled" }
  ]
}
```

Saving such an event needs the Flow Builder permission as well (see Privileges). Without them the call answers `403` with the missing flow privileges, and nothing is written:

```
{
  "errors": [{
    "status": "403",
    "code": "FRAMEWORK__MISSING_PRIVILEGE_ERROR",
    "detail": "{\"message\":\"Missing privilege\",\"missingPrivileges\":
[\"flow:create\",\"flow:update\",\"flow_sequence:create\",\"flow_sequence:update\"]}"
  }]
}
```

For a `business_event` trigger the extension creates and maintains a flow named **Event Counter: order_cancelled** with the action `action.sysea.event.counter.increase`. The flow's ID equals the event's ID; it is switched on and off with the event and removed when the event is deleted. You may add conditions to a managed flow; its trigger and action are rewritten whenever the event is saved. To count from a flow of your own, add the action *Increase event counter* in the Flow Builder and choose a server event.

```
{
  "id": "7d88e9dd112a4cdfaf45858a5280d3b9",
  "name": "Event Counter: order_cancelled",
  "eventName": "state_enter.order.state.cancelled",
  "active": true,
  "sequences": [
    {
      "actionName": "action.sysea.event.counter.increase",
      "config": { "definitionId": "7d88e9dd112a4cdfaf45858a5280d3b9", "technicalName": "order_cancelled" }
    }
  ]
}
```

Change, switch off, delete

```
PATCH  /api/sysea-event-counter-definition/{id}    { "active": false }
DELETE /api/sysea-event-counter-definition/{id}
```

Deleting an event keeps its counts. The report then shows the technical name as `event_label`. Creating an event with the same technical name later continues the same history.

Validation

Invalid writes answer `400` with one error per problem. `source.pointer` names the field.

```
{
  "errors": [
    {
      "code": "SYSEA_EVENT_COUNTER__TECHNICALNAMEINVALID",
      "status": "400",
      "detail": "sysea-event-counter.validation.technicalNameInvalid",
      "source": { "pointer": "/0/technicalName" }
    },
    {
      "code": "SYSEA_EVENT_COUNTER__TRIGGERTYPEINVALID",
      "status": "400",
      "detail": "sysea-event-counter.validation.triggerTypeInvalid",
      "source": { "pointer": "/0/triggers" }
    }
  ]
}
```

Code (`SYSEA_EVENT_COUNTER__…`)	Meaning
`TECHNICALNAMEINVALID`	Does not match `^[a-z][a-z0-9_]{2,63}$`
`TECHNICALNAMETAKEN`	Another event already uses the name
`TECHNICALNAMEIMMUTABLE`	The name of an existing event cannot be changed
`SCOPEINVALID`	Scope is neither `browser` nor `server`
`TRIGGERSINVALID`	`triggers` is not a list
`TOOMANYTRIGGERS`	More than 20 triggers
`TRIGGERTYPEINVALID`	Unknown type, or a type not allowed for the scope
`TRIGGERINCOMPLETE`	A required trigger field is missing or malformed
`ONEBUSINESSEVENT`	More than one `business_event` trigger on one event
`TRIGGERFIELDUNKNOWN`	A trigger contains a field its type does not have (see Trigger reference)

9. Trigger reference

Triggers are pure data. Nothing in them is executed, neither in the browser nor on the server. Each type accepts only the fields listed here; a trigger with any other field is rejected (`TRIGGERFIELDUNKNOWN`).

Type	Scope	Fields	Counts when
`dom_event`	browser	`eventName` , `selector`	An event with this name reaches an element matching the selector (see Storefront integration)
`page_view`	browser	`routes` (list) and/or `elementSelector`	The page's route is in the list and/or the element exists on the page. With both set, both must match.
`route`	server	`routes` (list, required)	Shopware answers a storefront request for one of these routes
`business_event`	server	`eventName`	The Flow Builder event fires (via the managed flow)
`core_event`	server	`key` : `cart_add` | `wishlist_add`	A product is added to the cart / to a customer's wishlist

```
// browser
{ "type": "dom_event", "eventName": "Wishlist/onProductAdded", "selector": "[data-wishlist-storage]" }
{ "type": "page_view", "routes": ["frontend.detail.page"] }
{ "type": "page_view", "elementSelector": ".cms-block-product-slider" }
// server
{ "type": "route", "routes": ["frontend.checkout.cart.page"] }
{ "type": "business_event", "eventName": "checkout.order.placed" }
{ "type": "core_event", "key": "cart_add" }
```

Field rules

- `eventName`: `^[A-Za-z][\w.:/-]{0,99}$`, e.g. `shown.bs.modal`, `Wishlist/onProductAdded`, `myshop:banner-shown`.
- `selector`, `elementSelector`: a CSS selector, at most 500 characters, no `<`. An invalid selector only disables its own trigger.
- `routes`: route names (`^[a-z0-9_.-]{1,128}$`) as listed by the catalog.

When server triggers count

- **Routes** count only for requests that reach Shopware. Pages served from the HTTP cache (`httpCache: true` in the catalog) are therefore not counted reliably; use a `page_view` browser trigger for them. A `GET` counts with a status below 300, a `POST` (or other method) with a status below 400, so a form post answered with a redirect counts. `HEAD` and `OPTIONS` never count.
- **Business events** count when a visitor caused them. They do not count when a merchant acts in the administration or an integration calls the Admin API, so an order created in the administration is not a shop order. Exception: state changes (`state_enter.*`, `state_leave.*`) always count, because payment providers report payments through webhooks.
- **Bots**: requests with a crawler, monitoring or script User-Agent are ignored for routes, core events, business events (except state changes) and browser batches.

---

10. Storefront integration

Theme and plugin developers count their own interactions by firing a DOM event and pointing a `dom_event` trigger at it. No API call and no JavaScript dependency on the extension is needed.

```
// In your theme or plugin JavaScript
const banner = document.querySelector('.myshop-banner');
banner.dispatchEvent(new CustomEvent('myshop:banner-shown'));
```

```
// The event definition (Administration › Marketing › Event counter, or via the API)
{
  "technicalName": "banner_shown",
  "name": "Banner shown",
  "active": true,
  "scope": "browser",
  "oncePerPage": true,
  "triggers": [{ "type": "dom_event", "eventName": "myshop:banner-shown", "selector": ".myshop-banner" }]
}
```

- The script registers **one capturing listener per event name on** `document`. It therefore also sees events that do not bubble, including the events of Shopware's JavaScript plugins: `this.$emitter.publish('Wishlist/onProductAdded')` in the wishlist storage plugin arrives under exactly that name.
- The target matches when `event.target.closest(selector)` finds an element. Use the selector `document` for events dispatched on `document` itself.

- Native events work the same way: `click` on `.btn-buy`, `submit` on `form.my-form`, Bootstrap's `shown.bs.modal` on a modal.
- Nothing is read from the page and nothing is stored in the browser. Only the technical names are sent.

Consent

Browser events are counted only after the visitor has accepted **"Anonymous usage statistics"** in Shopware's cookie banner (group *Statistics*, cookie `sysea-event-counter-enabled`). Accepting or withdrawing takes effect immediately, without a reload. If the shop switches off Shopware's own cookie banner and uses a third-party consent tool, the extension behaves like Shopware's Google Analytics integration: it no longer checks the cookie, and the consent tool is responsible. Server events do not use the cookie.

Transport

The script collects the events of a page and sends them in one request, at the latest 2 seconds after the last event and when the page is hidden (`fetch` with `keepalive`). A page view sends at most 20 requests, so even a frequent event such as `scroll` counted every time cannot flood the shop.

`POST` `/widgets/sysea-event-counter/batch` internal

```
{ "events": ["product_view", "wishlist_add", "wishlist_add"] }
```

This endpoint is internal to the storefront script and may change. It accepts only JSON requests from the shop's own pages (requests from other sites, recognised by their content type or `Sec-Fetch-Site` header, are ignored) and only active browser events of the current sales channel: at most 50 names per request and 20 per event, 1 for an event counted once per page. It ignores bots and always answers `204` so it reveals nothing about the configuration. Do not use it for imports or server-to-server counting; use a server event instead.

---

11. Recipes

curl

```
SHOP="https://shop.example.com"
TOKEN=$(curl -s -X POST "$SHOP/api/oauth/token" -H "Content-Type: application/json" \
  -d '{"grant_type":"client_credentials","client_id":"SWIA…","client_secret":"…"}' | jq -r .access_token)

curl -s -H "Authorization: Bearer $TOKEN" \
  "$SHOP/api/_action/sysea-event-counter/report?from=2026-09-01&to=2026-09-30&format=csv" \
  -o event-counter-2026-09.csv
```

Python

```
import requests

SHOP = "https://shop.example.com"
token = requests.post(f"{SHOP}/api/oauth/token", json={
    "grant_type": "client_credentials",
    "client_id": "SWIA…",
    "client_secret": "…",
}, timeout=30).json()["access_token"]

response = requests.get(
    f"{SHOP}/api/_action/sysea-event-counter/report",
    params={"from": "2026-10-01", "to": "2026-10-07", "technicalName[]": ["cart_add", "order_placed"]},
    headers={"Authorization": f"Bearer {token}", "Accept": "application/json"},
    timeout=60,
)
response.raise_for_status()
for row in response.json()["data"]:
    print(row["event_date"], row["event_name"], row["event_count"])
```

Power BI and Excel (Power Query)

```
let
    Shop = "https://shop.example.com",
    Token = Json.Document(Web.Contents(Shop, [
        RelativePath = "api/oauth/token",
        Headers = [#"Content-Type" = "application/json"],
        Content = Json.FromValue([grant_type = "client_credentials", client_id = "SWIA…", client_secret = "…"])
    ]))[access_token],
    Response = Web.Contents(Shop, [
        RelativePath = "api/_action/sysea-event-counter/report",
        Query = [from = "2026-01-01", format = "csv"],
        Headers = [Authorization = "Bearer " & Token]
    ]),
    Rows = Csv.Document(Response, [Delimiter = ",", Encoding = 65001, QuoteStyle = QuoteStyle.Csv]),
    WithHeaders = Table.PromoteHeaders(Rows, [PromoteAllScalars = true]),
    Typed = Table.TransformColumnTypes(WithHeaders, {{"event_date", type date}, {"event_count", Int64.Type}}, "en-US")
in
    Typed
```

Store the secret in a Power BI parameter rather than in the query. For scheduled refresh in the Power BI service, keep `Shop` as the base URL and pass the path through `RelativePath`, as shown.

Google Sheets and Looker Studio (Apps Script)

```
function importEventCounter() {
  const shop = 'https://shop.example.com';
  const props = PropertiesService.getScriptProperties();
  const token = JSON.parse(UrlFetchApp.fetch(shop + '/api/oauth/token', {
    method: 'post',
    contentType: 'application/json',
    payload: JSON.stringify({
      grant_type: 'client_credentials',
      client_id: props.getProperty('ACCESS_KEY'),
      client_secret: props.getProperty('SECRET_KEY'),
    }),
  }).getContentText()).access_token;

  const csv = UrlFetchApp.fetch(shop + '/api/_action/sysea-event-counter/report?format=csv&from=2026-01-01', {
    headers: { Authorization: 'Bearer ' + token },
  }).getContentText();

  const rows = Utilities.parseCsv(csv);
  const book = SpreadsheetApp.getActive();
  const sheet = book.getSheetByName('event_counter') || book.insertSheet('event_counter');
  sheet.clearContents();
  sheet.getRange(1, 1, rows.length, rows[0].length).setValues(rows);
}
```

Run it with a daily time-driven trigger. Looker Studio then reads the sheet with its Google Sheets connector. Google Analytics 4 cannot import aggregated daily counts, so connect the data to Looker Studio or BigQuery instead.

12. Errors

Status	When	Example
`400`	Invalid write to an event definition	See Validation
`401`	Missing, invalid or expired token	`"detail": "Missing \"Authorization\" header"`
`403`	The integration's role lacks a privilege	`"code": "FRAMEWORK__MISSING_PRIVILEGE_ERROR"`, `"detail": "{\"message\":\"Missing privilege\",\"missingPrivileges\":[\"order:read\"]}"`

Read endpoints never fail on parameter values: unknown dimensions, groupings and scopes fall back to their defaults, invalid dates to the default period, and invalid sales channel IDs are ignored.

13. Data, retention, privacy

- **Storage:** table `sysea_event_counter_daily`, unique per day, sales channel and technical name. Increments are atomic database upserts, so parallel requests never lose counts.

- **Retention:** configurable in the extension settings (`SyseaEventCounter.config.retentionMonths`), default 24 months. The scheduled task `sysea_event_counter.retention` runs daily and deletes older rows.
- **Switch per sales channel:** `SyseaEventCounter.config.active` (default on). When off, nothing is counted in that sales channel.
- **Privacy:** no personal data, no visitor profiles, no data sent to third parties. The only cookie is the consent cookie written by Shopware's cookie banner.
- **Uninstall:** with *Remove all app data permanently*, the extension removes its tables, its managed flows, its action in other flows and its settings. Otherwise the data stays for a later reinstall.

14. Changelog

Version	Date	Changes
1.0.1	2026-10	Event counter roles without flow privileges (Flow Builder events need the Flow Builder permission), CSV cells that look like formulas get a leading apostrophe, new validation code `TRIGGERFIELDUNKNOWN`, storefront endpoint accepts only the shop's own pages.
1.0.0	2026-10	First release: report (JSON/CSV), statistics, catalog, entities, storefront triggers, managed flows.

Event Counter for Shopware (SyseaEventCounter) · API reference for version 1.0.1 · This document describes the public interface. Endpoints marked "internal" may change without notice.