



GA4 Ecommerce Attributor for sGTM

Helps you enable Item list and Promotion attribution in GA4



Agenda

01 One Pager

02 Resources

03 What Ecom Attributor
can do for you?

04 Technical solution
explanation

05 Implementation guide

06 Calculating potential
Firestore cost

07 FAQ

01

One Pager

Ecommerce Attributor for sGTM



Customer Challenge

Advertisers often in their ecommerce implementation have List information only available when user clicks on the item inside the Item List. Similar is for Promotion - information is usually available only when user clicks on specific Promotion.

In order to get full data in GA4 Item List & Promotion reports, Item List and Promotion data needs to be sent with all ecommerce events.

Often, due to the complexity of a website, it is hard for advertisers to provide this information with every single ecommerce event, which creates a challenge how to make these reports actionable.



Solution Description

This solution is designed to pass List and Promotion data with every ecommerce event. Each time when user interacts with item or promotion (which contains List or Promotion information), List and Promotion information will be stored.

Depending on which version of solution you are using, information will be stored either in 1st party cookie or in Firestore. This data is then passed along with all subsequent ecommerce events to GA4.



Impact

Item List and Promotion attribution fully working in GA4, making Item List and Promotion reports fully populated with ecommerce data and ready to be used for analysis.



Limitations

Solution needs to be implemented in server-side Google Tag Manager container.
Data is stored in Firestore, which can generate additional GCP cost.

Requirements:



google/ga4-ecom-attributor

- Server-side GTM container up and running
- Access & billing enabled in Google Cloud Project
- List information available on select_item, view_item or add_to_cart event
- Item ID available on every ecommerce event

Item list name	+ ↓	Item-list view events	Item-list click events	Item list click-through rate	Add to baskets	Checkouts	E-commerce purchases	Item revenue
		27,137 100% of total	6,382 100% of total	47.31% Avg 0%	1,387 100% of total	2,351 100% of total	365 100% of total	€150,545.29 100% of total
Home - top products		10,543	312	5.58%	0	0	0	€0.00
Search results		4,554	2,246	57.41%	0	0	0	€0.00
On sale		3,943	308	15.68%	0	0	0	€0.00
You might also like		3,374	755	29.77%	0	0	0	€0.00
Top deals		2,505	1,382	58.61%	0	0	0	€0.00
Basket upsell		1,336	728	58.3%	0	0	0	€0.00
New products		540	208	46.79%	0	0	0	€0.00
(not set)		1	0	0%	1,387	1,999	364	€150,545.29



Item list name	+ ↓	Item-list view events	Item-list click events	Item list click-through rate	Add to baskets	Checkouts	E-commerce purchases	Item revenue
		3,725 100% of total	845 100% of total	47.41% Avg 0%	235 100% of total	398 100% of total	55 100% of total	€2,669.93 100% of total
Home - top products		1,364	40	6.47%	3	2	1	€3.32
Search results		696	317	61.26%	63	94	14	€506.09
On sale		638	143	35.34%	33	65	10	€382.86
You might also like		589	57	20.28%	9	41	4	€91.03
Top deals		178	106	58.33%	28	67	9	€391.79
Basket upsell		143	89	52.75%	12	24	4	€150.51
New products		59	33	61.76%	2	3	1	€10.47
(not set)		0	60	0%	83	194	28	€1,133.86

How to decide which version of solution is right for you?

	Ecom Attributor for web GTM (Github)	Ecom Attributor for sGTM (Github)
Ease of implementation	<ul style="list-style-type: none">● Implementation in web GTM container.● No changes required in the website code.	<ul style="list-style-type: none">● Implementation in sGTM container.● No changes required in the website code.● Requires GCP billing enabled and setup of Firestore database.
Requirements	<ul style="list-style-type: none">● Item ID present in each ecommerce event.	<ul style="list-style-type: none">● Item ID present in each ecommerce event.
Measurement support	<ul style="list-style-type: none">● Supports only Universal Analytics & GA4 Data Layer schema.● Works only if ecommerce measurement is implemented via Data Layer and web GTM (based on official Google ecommerce implementation documentation).● Data can still be sent to sGTM endpoint.	<ul style="list-style-type: none">● Solution supports all types of ecommerce measurement implementation (gtag.js, web GTM, third party tag management system, custom implementation).● Solution will work as long as ecommerce data received in sGTM respects GA4 ecommerce event data model (event and parameters naming).
Storage solution	<ul style="list-style-type: none">● List and Promotion information is stored in 1st party cookie.● Browser cookie limitations could apply (cross-domain measurement restriction, max number of cookies per domain, max cookie size etc.).	<ul style="list-style-type: none">● List and Promotion information is stored in Firestore.● Firestore quotas and limits apply.
Cost of Operation	<ul style="list-style-type: none">● No additional cost.	<ul style="list-style-type: none">● Additional cost could occur due to usage of sGTM and Firestore as a storage solution.

02

Resources

Resources



[group/ga4-ecom-attributor](#)



[google/ga4-ecom-attributor](#)

03

What Ecom Attributor can do for you?

Ecommerce measurement enables Item List & Promotion reports

With [GA4 ecommerce measurement](#), it is possible to send Item List and Promotion information.

Information will be populated in this two specific reports in GA4 and it enables you to analyze success of internal promotions and item lists on the website.

The screenshot shows the Google Analytics Monetization overview page. On the left, the navigation menu is expanded to show the 'Monetization' section, which is highlighted with a blue box. The main content area is titled 'Monetization overview' and contains several sections:

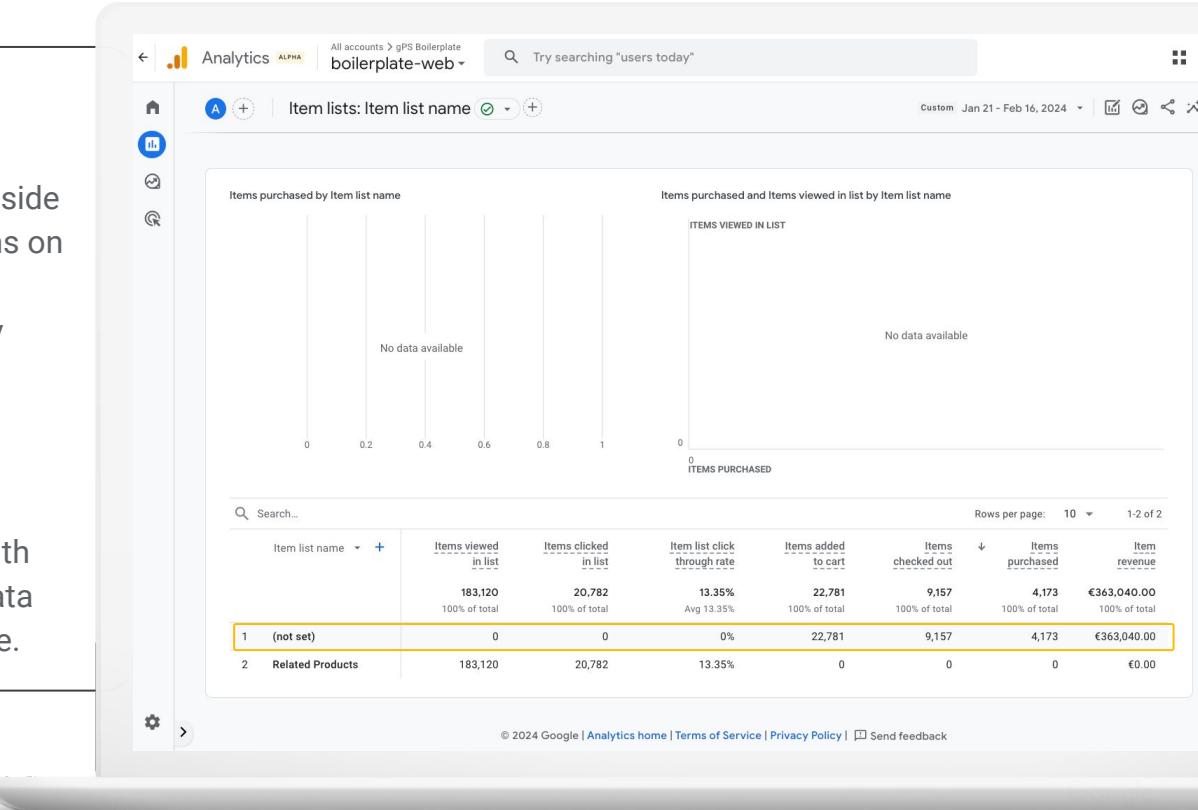
- Items purchased by item list name**: A table with columns 'ITEM LIST NAME' and 'ITEMS PURCHASED'. It shows 'No data available'.
- Items viewed in promotion by item promotion name**: A table with columns 'ITEM PROMOTION...' and 'ITEMS VIEWED...'. It shows 'No data available'.
- Purchase revenue by Order coupon**: A table with columns 'ORDER COUPON...' and 'PURCHASE REV...'. It shows 'No data available'.
- WHAT IS YOUR REVENUE FROM IN-APP PURCHASES?**: A table with columns 'PRODUCT ID' and 'PRODUCT REVENUE'. It shows 'No data available'.
- WHAT IS YOUR REVENUE PER AD UNIT?**: A table with columns 'AD UNIT' and 'PUBLISHER AD IMPRESSIO...'. It shows 'No data available'.

Item List report

How is report populated?

When user clicks on one of the items inside item list on the website (e.g. list of items on homepage that are on sale), Item List information needs to be sent with every ecommerce event in order to attribute Checkout and Revenue data to the appropriate Item List.

In case if List information is not sent with relevant ecommerce events, revenue data could be attributed to the (not set) value.

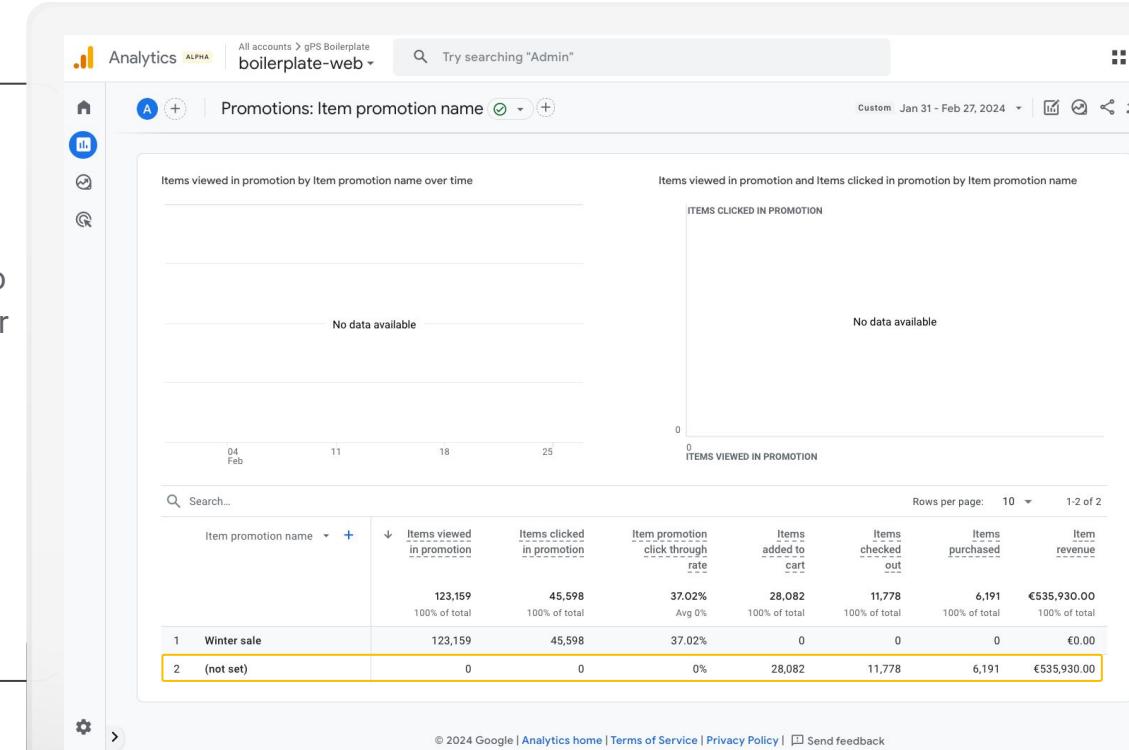


Promotions report

How is report populated?

When user clicks on one of promotions on the website, Promotion information needs to be sent with every ecommerce event in order to attribute Checkout and Revenue data to the appropriate Promotion.

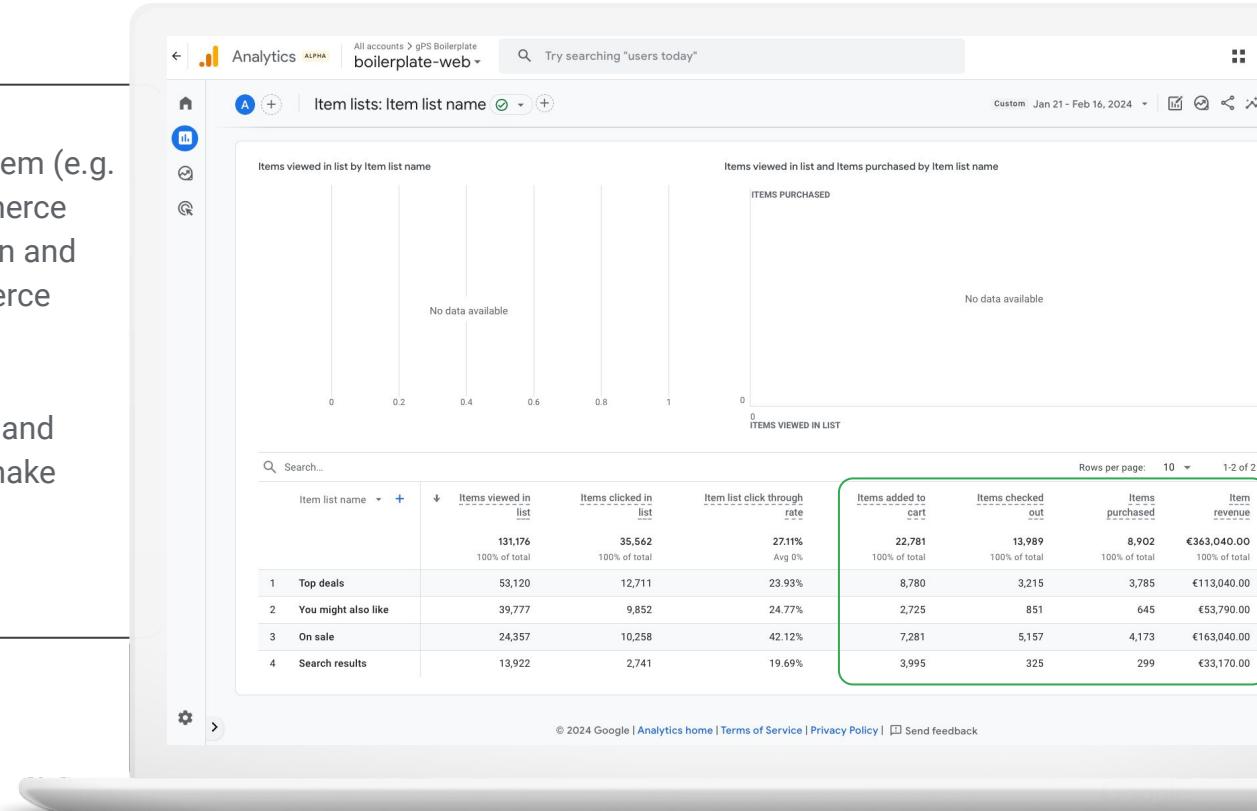
In case if Promotion information is not sent with relevant ecommerce events, revenue data could be attributed to the (not set) value.



Item List report with Ecommerce Attributor solution

Each time when user interacts with item (e.g. clicks on item inside the list), Ecommerce Attributor stores Item List information and sends it with all subsequent ecommerce events.

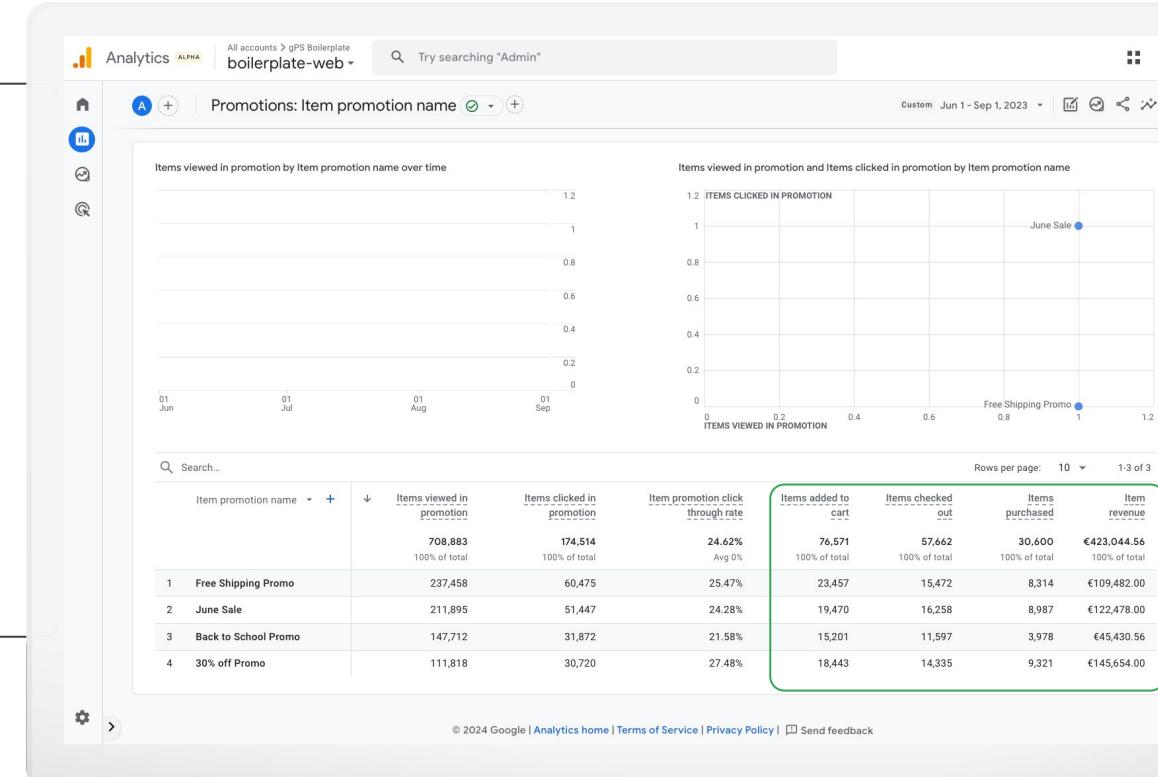
Data is populated in Item lists report and advertiser can start using report to make business decisions.



Promotions report with Ecommerce Attributor solution

Each time when user interacts with promotion (clicks on specific promotion on website), Ecommerce Attributor stores Promotion information and sends it with all subsequent ecommerce events.

Data is populated in GA4 Promotion report and advertiser can start using report to analyze Promotions and make business decisions.

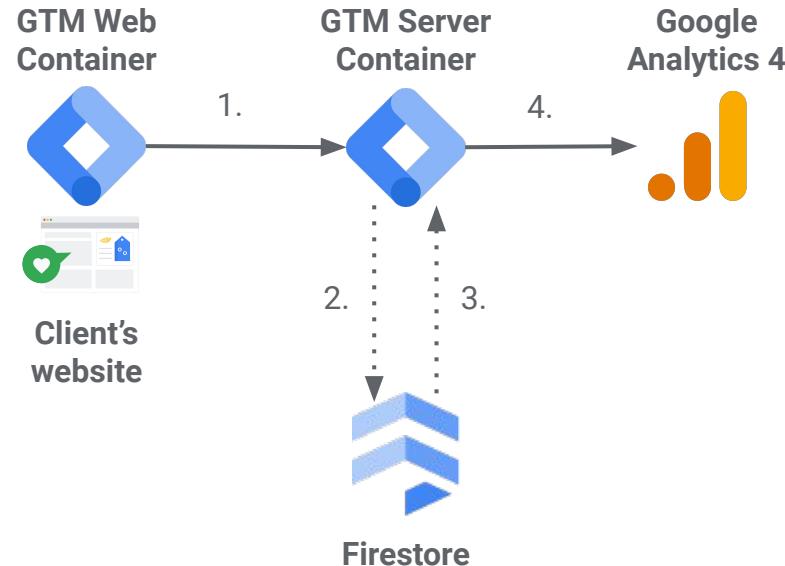


04

Technical solution explanation

Ecommerce Attributor for sGTM

1. User interacts with items on website, ecommerce event data is sent from website to server-side GTM container
2. If ecommerce event contains List and/or Promotion data, data is stored in Firestore for each item user had interaction with
3. When user progresses through the shopping funnel on your website (e.g. triggers ecommerce checkout events or purchase event), List and Promotion data is fetched from Firestore, appended to the item
4. Item data is sent to GA4, Checkout and Revenue data is attributed to the appropriate List and Promotion



05

Implementation guide

Pre-requisites



Server Side Google Tag Manager

Deployed sGTM container and website data is routed to the sGTM endpoint



Google Cloud Project

Access to a Google Cloud project with Firestore in [Native mode](#)



Ecommerce measurement

Active ecommerce measurement implemented on website with List and/or Promotion data

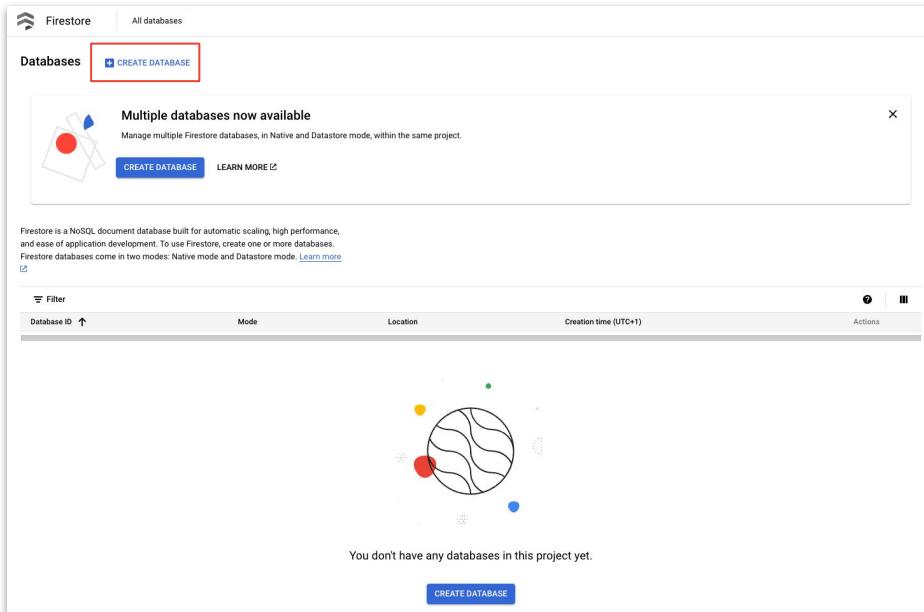
Create Firestore Database

Step 1

Go to Google Cloud Platform, to the Firestore page.

If you don't have any databases created, click on **Create database** button.

If you already have database created, make sure you have Database with name **(default)**, Native mode.



Create Firestore Database

Step 2

In first step, for Firestore mode select **Native mode (recommended)** and click on Continue.

In second step, keep Database ID as it is (don't change it): **(default)**

For location type, select what you prefer. In this example, we will select Region and europe-west3.

Once you are finished, click on blue button **Create database**.

1 Select your Firestore mode
Select between Native or Datastore mode

2 Configure your database

Native mode (recommended)
Recommended for all servers, mobile apps, and web apps

Datastore mode
Use Datastore mode if your app requires the Datastore API

COMPARE MODES

CONTINUE

1 Select your Firestore mode
Native mode

2 Configure your database

Name your database
Permanent choice
Database ID *
(default)

Location
Permanent choice: Determines where your computing resources and data are located. Affects cost, performance and replication. [Learn More](#)

Location type
 Region
99.99% availability SLA, lower latency within a single region

Multi-region
99.999% availability SLA, highest availability across largest area

Some locations have been restricted due to a policy set by your organization. [Learn more about restricting locations](#)

Region *
europe-west3 (Frankfurt)

Secure rules
Security rules provide access control and data validation. Start with one of two preset rules and customize these later.

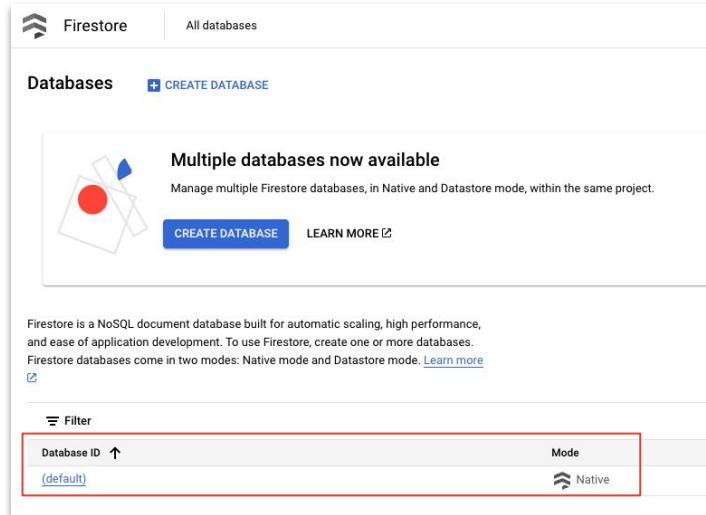
Production rules
Your data is private by default. Mobile and web client library access will only be granted as specified by your security rules.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```

Create Firestore Database

Overview

After Firestore Database is created, it should look like this.



The screenshot shows the Firestore console interface. At the top, there's a header with the Firestore logo and a link to 'All databases'. Below the header, a section titled 'Databases' features a 'CREATE DATABASE' button and a 'LEARN MORE' link. A large image of overlapping documents with a red dot on one is displayed. A descriptive text block states: 'Multiple databases now available. Manage multiple Firestore databases, in Native and Datastore mode, within the same project.' Below this, a detailed description of Firestore is provided, mentioning its NoSQL document database nature, automatic scaling, high performance, and ease of application development. It also notes that databases come in two modes: Native mode and Datastore mode, with a 'Learn more' link. At the bottom, a table titled 'Filter' lists a single database entry: '(default)' under 'Database ID' and 'Native' under 'Mode'. The entire table row is highlighted with a red border.

Database ID	Mode
(default)	Native

Create Time-to-live (TTL) policy

Step 1

In Firestore, click on **(default)** database, and then on left-side navigation, click on **Time-to-live (TTL)**.

Click on **Create Policy**.

As **Collection group** enter: **ecommerce_attributor**
As **Timestamp field**, enter: **expiration**

Click on **Save** button. It can take some time until policy is created.

The screenshot shows the Firestore console interface. On the left, there's a sidebar with options like 'Database', 'Indexes', 'Import/Export', 'Disaster Recovery', and 'Time-to-live (TTL)'. The 'Time-to-live (TTL)' option is highlighted with a red box. On the right, under 'Time-to-live (TTL)', there's a 'CREATE POLICY' button also highlighted with a red box. Below this, a descriptive text says 'TTL policies are used to schedule expiration and bulk deletion jobs run continuously in the background.' followed by a 'Learn more' link. There's also a 'Filter' button and a 'Collection group' dropdown. The main area says 'No policies to display'.

The screenshot shows the 'Create a TTL policy' form. It has a heading 'Create a TTL policy' and a note 'You can set one TTL policy per collection group.' followed by a 'Learn more' link. Below this is a section 'Enter a timestamp field' with a note 'Documents are scheduled for deletion based on a timestamp field in the specified collection group. Data is typically deleted within 72 hours after its expiration date.' A warning icon with a note 'Single-field indexes may cause issues with the field receiving a TTL.' is shown, followed by a 'Learn more' link. At the bottom, there are two input fields: 'Collection group *' containing 'ecommerce_attributor' and 'Timestamp field *' containing 'expiration', both highlighted with red boxes.

Create Time-to-live (TTL) policy

Overview

Once TTL policy is created, it should look like this.

The screenshot shows the Firestore console interface. On the left, there's a sidebar with icons for Database, Firestore Studio, Indexes, Import/Export, Disaster Recovery, Time-to-live (TTL), and Security Rules. The 'Time-to-live (TTL)' item is highlighted with a blue background. The main area has a breadcrumb navigation: All databases > DATABASE (default). A red box highlights the 'DATABASE (default)' link. Below the breadcrumb, there's a section titled 'Time-to-live (TTL)' with a '+ CREATE POLICY' button. A descriptive text states: 'TTL policies are used to schedule expiration and bulk deletion of documents. Deletion jobs run continuously in the background.' A 'Learn more' link is provided. A 'Filter' button is available to refine the table. The table lists a single policy: 'Collection group' is 'ecommerce_attributor', 'Timestamp field' is 'expiration', and 'Status' is 'Serving'. A red box highlights the 'Status' column header.

Collection group	Timestamp field	Status
ecommerce_attributor	expiration	Serving

Provide sGTM access to Firestore project

This step is necessary only for users who have two separate GCP projects for sGTM container and for Firestore database.

In case if your sGTM container and Firestore database are located in the same Google Cloud Platform project, you can skip this step.



If you have GCP setup just like it is shown on picture, follow steps on next slide.

This is required in order to give server-side GTM access to the Firestore project.

Provide sGTM access to Firestore project

Step 1

Navigate to your GCP project where you have **server-side GTM container deployed**.

Go to **IAM & Admin** and in left-side navigation, click on **Service Accounts**.

Copy your service account email address.

The screenshot shows the GCP IAM & Admin interface. On the left, there's a sidebar with links: IAM, Identity & Organization, Policy Troubleshooter, Policy Analyzer (NEW), Organization Policies, Service Accounts (which is highlighted with a red box), and Workload Identity Federation. The main area has a header with 'Service accounts', a 'CREATE SERVICE ACCOUNT' button, and a 'DELETE' button. Below the header, it says 'Service accounts for project "boilerplate-sgtm"' and provides a brief description of what service accounts are. A 'Filter' input field is present. The table lists one service account: an email ending in '@appspot.gserviceaccount.com' with a status of 'Enabled'. A 'Copy to clipboard' button is located at the bottom right of this row. The entire 'Service Accounts' section is also highlighted with a red box.

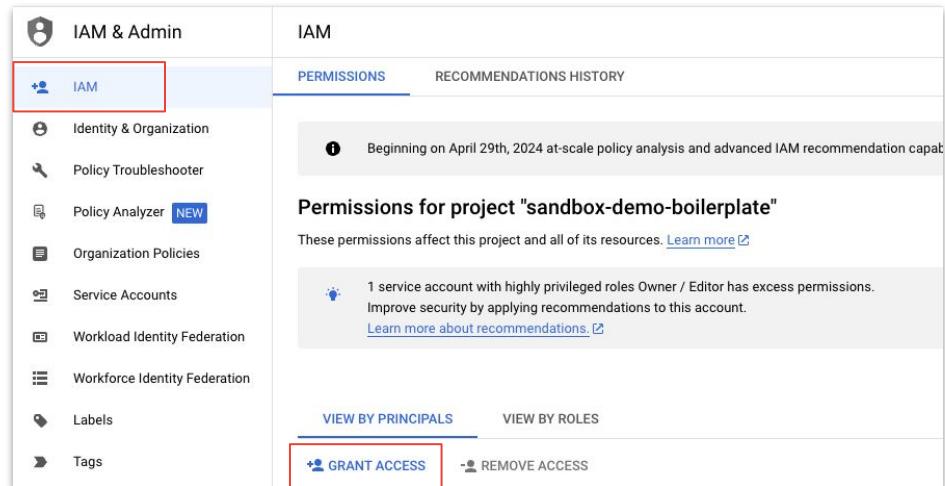
Provide sGTM access to Firestore project

Step 2

Now, navigate to your GCP project which you will use to store data in Firestore (GCP project where you created Database and TTL policy in Firestore in previous step).

Go to **IAM & Admin** and in left-side navigation, click on **IAM**.

Click on **Grant Access**.



Provide sGTM access to Firestore project

Step 3

Paste your service account in **New Principals** field.

As a Role, select **Cloud Datastore User**.

Click **Save** button.

The screenshot shows two stacked dialog boxes from the Google Cloud IAM interface.

Add principals (Top Dialog):

- Header: "Add principals".
- Description: "Principals are users, groups, domains, or service accounts. [Learn more about principals in IAM](#)".
- Input field: "New principals * —" containing a placeholder email address: "[REDACTED]@appspot.gserviceaccount.com".
- Buttons: A question mark icon and a close/cancel icon.

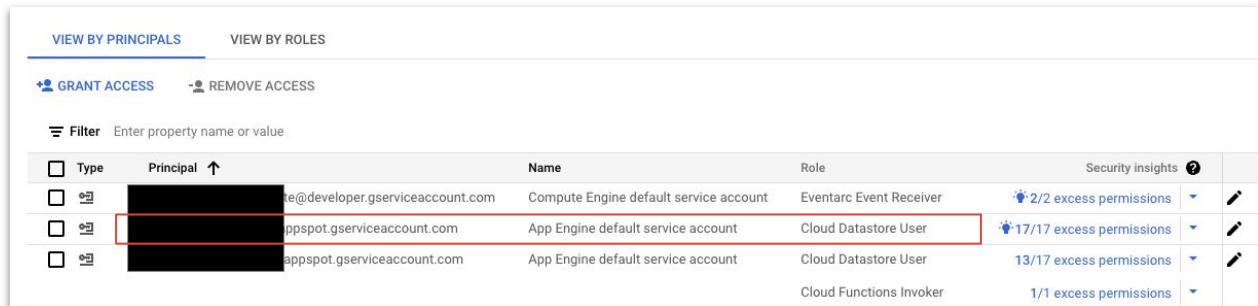
Assign roles (Bottom Dialog):

- Header: "Assign roles".
- Description: "Roles are composed of sets of permissions and determine what the principal can do with this resource. [Learn more](#)".
- Role selection: "Role * —" dropdown set to "Cloud Datastore User".
- Condition: "IAM condition (optional)" section with a "Cloud Datastore User" entry and a "+ ADD IAM CONDITION" link.
- Description: "Provides read/write access to data in a Cloud Datastore database. Intended for application developers and service accounts."
- Buttons: "+ ADD ANOTHER ROLE", "SAVE" (blue button), and "CANCEL".

Provide sGTM access to Firestore project

Overview

After you provide access, you should see your Service account email listed with Cloud Datastore User access.



The screenshot shows the Google Cloud IAM interface for granting access to a project. The interface has two main tabs at the top: 'VIEW BY PRINCIPALS' (selected) and 'VIEW BY ROLES'. Below these are buttons for 'GRANT ACCESS' and 'REMOVE ACCESS'. A 'Filter' input field is present. The main area is a table listing access grants:

Type	Principal	Name	Role	Security insights	Action
<input type="checkbox"/>	[REDACTED]	te@developer.gserviceaccount.com	Compute Engine default service account	Eventarc Event Receiver 2/2 excess permissions	
<input type="checkbox"/>	[REDACTED]	ppspot.gserviceaccount.com	App Engine default service account	Cloud Datastore User 17/17 excess permissions	
<input type="checkbox"/>	[REDACTED]	appspot.gserviceaccount.com	App Engine default service account	Cloud Datastore User 13/17 excess permissions	
			Cloud Functions Invoker	1/1 excess permissions	

Prerequisites before importing solution in sGTM

Before you import JSON file with solution, go to your sGTM container and make sure you have:

- Built-in Variable **Event Name** enabled
- GA4 event tag that forwards data to GA4

Built-In Variables ⓘ	
Name ↑	Type
Client Name	Client Name
Container ID	Container ID
Container Version	Container Version Number
Debug Mode	Debug Mode
Event Name	Event Name
Query String	Query String

Tags		
<input type="checkbox"/>	Name ↓	Type
<input type="checkbox"/>	GA4 All Events	Google Analytics: GA4 

sGTM JSON import

Step 1

Join [Google Group](#), to be able to download JSON file and to get notifications about solution updates.

Download the [JSON file](#) (available on [Github](#)).



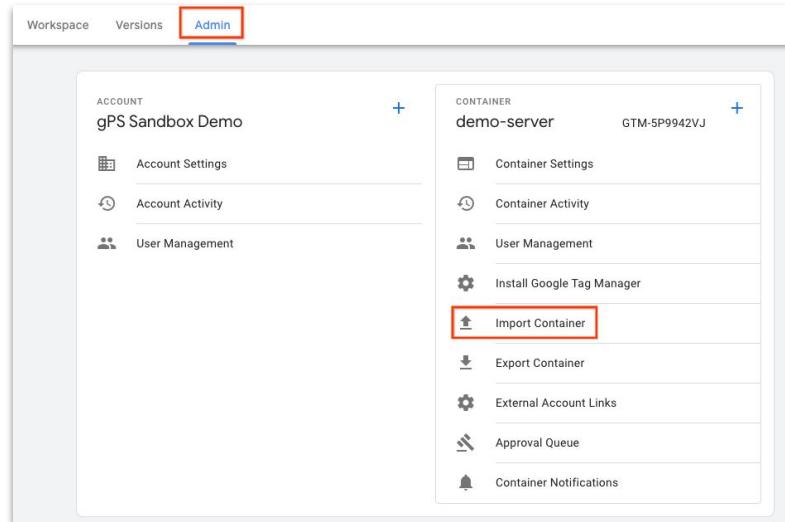
Make sure it is saved as **.json** file, otherwise it will not work.

The file is exported sGTM container, which contains tags, variables and custom built templates necessary for this solution to work.

sGTM JSON import

Step 2

1. Go to Google Tag Manager
2. Open your server-side GTM container where you want to implement solution
3. Click **Admin → Import Container**



sGTM JSON import

Step 3

1. Select downloaded JSON file to import
2. Select in which workspace you want to import the solution
3. Select **Merge** and then **Rename conflicting tags, triggers and variables**
4. Click on **Confirm** button

Overwrite or merge with the latest container version by importing a json file in the correct format.

Select file to import

Choose workspace

Choose an import option (?)

Overwrite
Overwrite selected workspace with content of imported container GTM-5P9942VJ

Merge
Merge selected workspace with content of imported container GTM-5P9942VJ

Overwrite conflicting clients, tags, transformations, triggers and variables.

Rename conflicting clients, tags, transformations, triggers and variables.

Preview and confirm your import:

Clients	Transformations	Tags	Triggers
0 0 0	2 0 0	1 0 0	0 0 0
New Modified Deleted	New Modified Deleted	New Modified Deleted	New Modified Deleted

Variables	Templates
10 0 0	3 0 0
New Modified Deleted	New Modified Deleted

[View Detailed Changes](#)

sGTM JSON import

Overview

After you import JSON file, you should see these changes in your workspace.

Next, we need to set up following things:

- Add your GCP Project ID in **Constant - GCP Project ID** variable
- Configure **Ecom Attributor - Write to Firestore** tag
- Configure triggers for **Ecom Attributor - Write to Firestore** tag
- Optional: configure **Identifier - hashed** variable

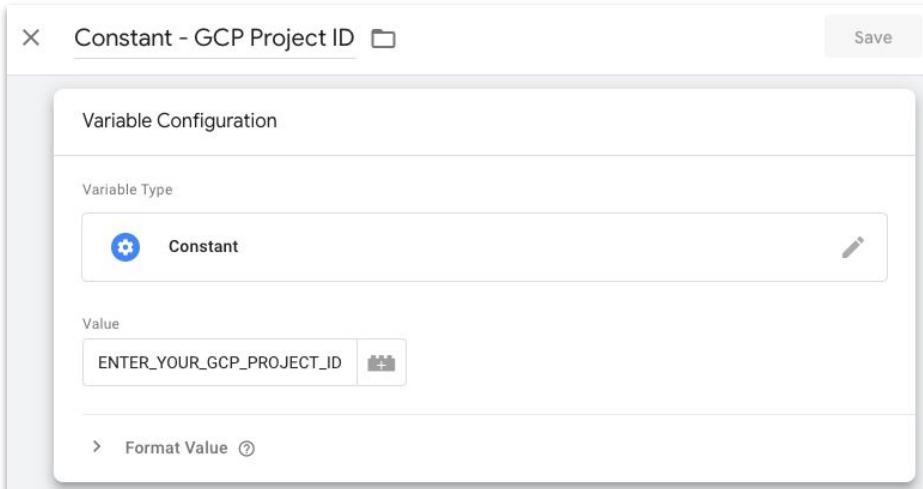
Workspace Changes		
Name	Type ↑	Change
Ecom Attributor - Identifier hasher	Custom Template	Added
Ecom Attributor - Items array	Custom Template	Added
Ecom Attributor - Write to Firestore	Custom Template	Added
Ecom Attributor	Folder	Added
Ecom Attributor - Write to Firestore	Tag	Added
Ecom Attributor - Augment - Item List data	Transformation	Added
Ecom Attributor - Augment - Promotion data	Transformation	Added
Identifier - Event Data - client_id	Variable	Added
Identifier - hashed	Variable	Added
Constant - FS Collection Path	Variable	Added
Constant - GCP Project ID	Variable	Added
FS Lookup - promotion_id	Variable	Added
FS Lookup - attribution	Variable	Added
Ecom Attributor - Items array	Variable	Added
FS Lookup - creative_name	Variable	Added
FS Lookup - promotion_name	Variable	Added
FS Lookup - creative_slot	Variable	Added

Configure Constant - GCP Project ID variable

Open variable called **Constant - GCP Project ID**

Delete everything from the input field, and **provide your Google Cloud Project ID**.

NOTE: This should be your GCP Project ID where you configured Firestore database in previous steps.



Configure Ecom Attributor - Write to Firestore tag

Step 1

Open tag called **Ecom Attributor - Write to Firestore**

In the tag, you don't have to configure any GCP data or Identifier variable (this is optional, it will be explained in later steps).

Scroll down to the section **Which information do you want to collect and store in Firestore.**

Mark checkboxes for which Attribution type you want to use this solution:

- Item List Attribution
- Promotion Attribution

Which information do you want to collect and store in Firestore [?](#)

- Item List Attribution [?](#)
- Promotion Attribution [?](#)

Configure Ecom Attributor - Write to Firestore tag

Step 2

Next, go to section called **Attribution time**.

Configure how long you want to keep data in Firestore. By default it is 7 days. This means if user interacts with Item inside the List or with Promotion, List and Promotion data will be kept in Firestore for 7 days after user's last ecommerce interaction. In case if user inside the 7 days window makes a purchase of item which he had interaction with, List and Promotion data from Firestore will be sent to GA4 and Revenue data will be properly attributed.

NOTE: When data is written in Firestore, there is field called **expiration**, which is timestamp set in the future (by default 7 days in the future). Expiration field is used by Firestore TTL policy, to automatically delete user data after 7 days, if Item/Promotion data wasn't refreshed in Firestore during those 7 days.

Decreasing the attribution time (e.g. to only 1 day) can increase Firestore costs due to more frequent TTL policy deletes.

Attribution time ⓘ

Configure how long you want to keep data in Firestore for each user (expiration field in Firestore). By default it is 7 days. This means if user interacts with Item inside the List or with Promotion, List and Promotion data will be kept in Firestore for 7 days after user's last ecommerce interaction. In case if user during the 7 days window makes a purchase of item which he had interaction with, List and Promotion data from Firestore will be sent to GA4 and Revenue data will be properly attributed.

Expiration field is used by Firestore TTL policy, to automatically delete user data after 7 days (default setting), if Item/Promotion data wasn't refreshed in Firestore during those 7 days.

Note: decreasing the attribution time (e.g. to only 1 day), could increase Firestore costs due to the more frequent number of TTL policy deletes.

If you want to have shorter/longer expiration date, adjust number of days in input field. If you want to keep it as default, provide number 7 as input.



Configure Ecom Attributor - Write to Firestore tag

Step 3

Next, go to section called **Delete List and Promotion data on purchase event.**

What happens if you turn on this option:

When user makes a purchase, if user purchased itemA and itemB, and if there is Promotion and List data stored in Firestore for these two items, this information will be deleted from Firestore.

NOTE: In case if user has List data available for items which he didn't purchase currently, this information will stay in Firestore.

Delete List and Promotion data on purchase event (recommended) 

Remove List and Promo data for purchased items from Firestore on purchase event

Configure Ecom Attributor - Write to Firestore tag

Overview

After you configure all settings, this is how Ecom Attributor - Write to Firestore tag should look like.

Depending on which settings you selected, your setup will be of course slightly different compared to this example.

The screenshot shows the configuration interface for the Ecom Attributor - Write to Firestore tag. It is divided into several sections:

- Google Cloud Project data**:
 - GCP Project ID: {{Constant - GCP Project ID}}
 - Firestore Collection Path: {{Constant - FS Collection Path}}
 - Firestore Lookup Variable: {{FS Lookup - attribution}}
- Hashed identifier used for storing data per user/device**:
 - Hashed identifier: {{Identifier - hashed}}
- Which information do you want to collect and store in Firestore**:
 - Item List Attribution
 - Promotion Attribution
- Attribution time**:

Configure how long you want to keep data in Firestore for each user (expiration field in Firestore). By default it is 7 days. This means if user interacts with Item inside the List or with Promotion, List and Promotion data will be kept in Firestore for 7 days after user's last ecommerce interaction. In case if user during the 7 days window makes a purchase of item which he had interaction with, List and Promotion data from Firestore will be sent to GA4 and Revenue data will be properly attributed.

Expiration field is used by Firestore TTL policy, to automatically delete user data after 7 days (default setting), if Item/Promotion data wasn't refreshed in Firestore during those 7 days.

Note: decreasing the attribution time (e.g. to only 1 day), could increase Firestore costs due to the more frequent number of TTL policy deletes.

If you want to have shorter/longer expiration date, adjust number of days in input field. If you want to keep it as default, provide number 7 as input.

7
- Delete List and Promotion data on purchase event (recommended)**:
 - Remove List and Promo data for purchased items from Firestore on purchase event

Add triggers to Ecom Attributor - Write to Firestore tag

Step 1

Next, we need to provide triggers for **Ecom Attributor - Write to Firestore tag**.

If you selected option **Item List Attribution**, you need to know on which event you have List information available in Event Data model. Solution supports collection of List data on following events:

- select_item
- view_item
- add_to_cart

NOTE: If you have cases on website when List data is available on all three mentioned ecommerce events, solution supports this as well. Just add all three events as a trigger to the tag.

If List data is available outside of items array, solution supports this as well and it will collect List information.

If Promotion data is available with Item, this will be collected and stored in Firestore as well.



```
gtag("event", "select_item", {
  items: [
    {
      item_id: "SKU_12345",
      item_name: "Stan and Friends Tee",
      item_brand: "Google",
      item_category: "Apparel",
      item_category2: "Adult",
      item_category3: "Shirts",
      item_list_id: "related_products",
      item_list_name: "Related Products",
      location_id: "ChIJIQBpAG2ahYAR_6128GcTUEo",
      index: 1,
      promotion_id: "P_12345",
      promotion_name: "Summer Sale",
      creative_name: "summer_banner2",
      creative_slot: "featured_app_1"
    }
  );
});
```

Add triggers to Ecom Attributor - Write to Firestore tag

Step 1

If you selected **Item List Attribution** option, and if you have List data available only on **select_item** event, add this event as a trigger to the tag.

If you have case where List information is also available on **view_item** and **add_to_cart** event, add those events are trigger as well.

The screenshot shows the configuration interface for a 'Write to Firestore' tag. It includes sections for collecting information, setting attribution time, deleting data on purchase, and defining triggers.

- Which information do you want to collect and store in Firestore:** A checkbox for 'Item List Attribution' is checked and highlighted with a red border.
- Attribution time:** A note explaining the default 7-day TTL for user interactions with items in a list or promotion.
- Expiration field:** A note about the TTL policy used by Firestore.
- Note:** A note about decreasing attribution time increasing Firestore costs.
- Expiration input field:** A numeric input set to '7' with a calendar icon.
- Delete List and Promotion data on purchase event (recommended):** A checkbox for removing data from Firestore on purchase.
- Triggering:** A section for defining triggers, showing a list with 'select_item' highlighted with a red border.

Add triggers to Ecom Attributor - Write to Firestore tag

Step 2

If you selected **Promotion Attribution** option, you have to provide **select_promotion** trigger to the tag.

NOTE: Solution supports collection of Promotion data on select_promotion event only.

Solution will collect Promotion data if it is available inside items array but also it supports case if you are sending it outside of array.

Which information do you want to collect and store in Firestore [?](#)

Item List Attribution [?](#)

Promotion Attribution [?](#)

```
gtag("event", "select_promotion", {  
  creative_name: "Summer Banner",  
  creative_slot: "featured_app_1",  
  promotion_id: "P_12345",  
  promotion_name: "Summer Sale",  
  items: [  
    item_id: "SKU_12345",  
    item_name: "Stan and Friends Tee",  
    item_brand: "Google",  
    item_category: "Apparel",  
    item_category2: "Adult",  
    item_category3: "Shirts",  
    item_list_id: "related_products",  
    item_list_name: "Related Products",  
    location_id: "ChIJIQBpAG2ahYAR_6128GcTUEo",  
    index: 1  
  ]  
});
```

Add triggers to Ecom Attributor - Write to Firestore tag

Step 2

If you selected **Promotion Attribution** option, provide **select_promotion** event as a trigger to the tag.

The screenshot shows the configuration interface for the 'Write to Firestore tag'. It includes sections for collecting information, setting attribution time, deleting data on purchase, and defining triggers.

Which information do you want to collect and store in Firestore

Item List Attribution

Promotion Attribution (highlighted with a red box)

Attribution time

Configure how long you want to keep data in Firestore for each user (expiration field in Firestore). By default it is 7 days. This means if user interacts with item inside the List or with Promotion, List and Promotion data will be kept in Firestore for 7 days after user's last ecommerce interaction. In case if user during the 7 days window makes a purchase of item which he had interaction with, List and Promotion data from Firestore will be sent to GA4 and Revenue data will be properly attributed.

Expiration field is used by Firestore TTL policy, to automatically delete user data after 7 days (default setting), if item/Promotion data wasn't refreshed in Firestore during those 7 days.

Note: decreasing the attribution time (e.g. to only 1 day), could increase Firestore costs due to the more frequent number of TTL policy deletes.

If you want to have shorter/longer expiration date, adjust number of days in input field. If you want to keep it as default, provide number 7 as input.

7

Delete List and Promotion data on purchase event (recommended)

Remove List and Promo data for purchased items from Firestore on purchase event

Triggering

Firing Triggers

select_item
Custom Event

select_promotion
Custom Event (highlighted with a red box)

Add triggers to Ecom Attributor - Write to Firestore tag

Step 3

If you selected option **Remove List and Promo data for purchased items**, provide purchase event as a trigger to the tag.

The screenshot shows the configuration interface for the 'Write to Firestore' tag. It includes sections for collecting information, setting attribution time, and managing data deletion on purchase events. The 'purchase' trigger is selected under 'Triggering'.

- Which information do you want to collect and store in Firestore:**
 - Item List Attribution
 - Promotion Attribution
- Attribution time:**

Configure how long you want to keep data in Firestore for each user (expiration field in Firestore). By default it is 7 days. This means if user interacts with Item inside the List or with Promotion, List and Promotion data will be kept in Firestore for 7 days after user's last ecommerce interaction. In case if user during the 7 days window makes a purchase of item which he had interaction with, List and Promotion data from Firestore will be sent to GA4 and Revenue data will be properly attributed.
- Expiration field:** Expiration field is used by Firestore TTL policy, to automatically delete user data after 7 days (default setting), if Item/Promotion data wasn't refreshed in Firestore during those 7 days.
- Note:** decreasing the attribution time (e.g. to only 1 day), could increase Firestore costs due to the more frequent number of TTL policy deletes.
- If you want to have shorter/longer expiration date, adjust number of days in input field. If you want to keep it as default, provide number 7 as input.**
- Delete List and Promotion data on purchase event (recommended):**
 - Remove List and Promo data for purchased items from Firestore on purchase event
- Triggering:**

Firing Triggers

 - purchase Custom Event
 - select_item Custom Event
 - select_promotion Custom Event

Add triggers to Ecom Attributor - Write to Firestore tag

Overview

Depending on which setting you selected, you should have:

- If you selected **Item List Attribution**, you should have **at least 1 trigger** (select_item, view_cart or add_to_cart)
- In case if you selected **Promotion Attribution**, you should have **select_promotion event as trigger**
- If you selected to **Remove List and Promo data after purchase**, you should have **purchase event as trigger**

IMPORTANT: If you have advanced Consent Mode implemented, it is worth to note that this solution won't work for unconsented pings. We recommend to **not trigger** Write to Firestore tag on unconsented pings.

The screenshot shows the configuration interface for the "Write to Firestore tag" trigger in Ecom Attributor. It includes sections for collecting information, attribution time, expiration field, and advanced settings, along with a triggering section defining the firing triggers.

Which information do you want to collect and store in Firestore

Item List Attribution

Promotion Attribution

Attribution time

Configure how long you want to keep data in Firestore for each user (expiration field in Firestore). By default it is 7 days. This means if user interacts with Item inside the List or with Promotion, List and Promotion data will be kept in Firestore for 7 days after user's last ecommerce interaction. In case if user during the 7 days window makes a purchase of item which he had interaction with, List and Promotion data from Firestore will be sent to GA4 and Revenue data will be properly attributed.

Expiration field is used by Firestore TTL policy, to automatically delete user data after 7 days (default setting), if Item/Promotion data wasn't refreshed in Firestore during those 7 days.

Note: decreasing the attribution time (e.g. to only 1 day), could increase Firestore costs due to the more frequent number of TTL policy deletes.

If you want to have shorter/longer expiration date, adjust number of days in input field. If you want to keep it as default, provide number 7 as input.

Delete List and Promotion data on purchase event (recommended)

Remove List and Promo data for purchased items from Firestore on purchase event

Advanced Settings

Triggering

Firing Triggers

purchase
Custom Event

OR

select_item
Custom Event

OR

select_promotion
Custom Event

Optional #1: Configure Identifier - hashed variable

This step is optional.

By default, solution is using hashed **client_id** to store data in Firestore (SHA-256, encoded in hex).

In case if you want to use some other device/user identifier, then open variable called **Identifier - hashed**. By default, as an input client_id variable is provided. Remove it, and provide variable with identifier you would like to use.

IMPORTANT: if you want to use some other device/user identifier (e.g. user_id), this identifier has to be available on every ecommerce event, otherwise solution won't work.

The screenshot shows the 'Identifier - hashed' configuration screen. At the top, there's a title bar with the name of the variable. Below it is a 'Variable Configuration' section. Under 'Variable Type', it shows 'Ecom Attributor - Identifier hasher'. There's also a 'Variable permissions' section with a note about 1 permission. A descriptive note below states that item & promo data is stored in Firestore for each user, and it requires a user/device identifier. It recommends using client_id by default but notes that using user_id or other identifiers requires them to be available on every ecommerce event. An 'Encoding' section explains that it calculates and returns the SHA-256 digest of the input, encoded in hex. A text input field is provided for the variable to hash, with '(Identifier - Event Data - client_id)' as the current value. A 'Format Value' button is located at the bottom right.

Optional #2: Remove Promotion Transformation & Variables

This step is optional and should only be done if you don't have Promotions measurement implemented on your website

In case if you don't have promotion measurement implemented on your website and you are not storing any Promotion information in Firestore, you can delete following things from your sGTM container:

- Ecom Attributor - Augment - Promotion data
- FS Lookup - promotion_name
- FS Lookup - promotion_id
- FS Lookup - creative_slot
- FS Lookup - creative_name

Transformations		Type
<input type="checkbox"/>	Name ↑	
<input type="checkbox"/>	Ecom Attributor - Augment - Item List data	Augment event
<input type="checkbox"/>	Ecom Attributor - Augment - Promotion data	Augment event
User-Defined Variables		
<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	FS Lookup - promotion_name	Firestore Lookup
<input type="checkbox"/>	FS Lookup - promotion_id	Firestore Lookup
<input type="checkbox"/>	FS Lookup - creative_slot	Firestore Lookup
<input type="checkbox"/>	FS Lookup - creative_name	Firestore Lookup

Summary

Final checklist what you should have done using this step-by-step walkthrough

01

Google Cloud Platform setup

- Create (default) Firestore database (Native mode)
- Create TTL policy for **expiration** field in **ecommerce_attributor** collection

02

Import JSON file & Configure Constant - GCP Project ID variable

- Join [Google Group](#)
- Download [JSON](#) file from Github & import in your sGTM container
- Provide your GCP Project ID in Constant - GCP Project ID variable

03

Configure settings in Ecom Attributor - Write to Firestore tag

- Which information you want to collect in Firestore: Item List and/or Promotion data
- How long you want to keep user data in Firestore (default 7 days)
- Do you want to remove List & Promo data for purchased items from Firestore after purchase event

04

Add triggers to the Ecom Attributor - Write to Firestore tag

- If you selected to collect Item List data, you need to provide at least 1 trigger (select_item, view_item and/or add_to_cart)
- If you selected to collect Promotion data, you need to provide select_promotion event as trigger
- If you selected to remove List & Promo data after purchase event, you need to provide purchase event as trigger

How to test: Item List attribution (case No.1)

```
gtag("event", "select_item", {  
  items: [{  
    item_id: "SKU_123",  
    item_name: "Stan and Friends Tee",  
    item_brand: "You Tube",  
    item_category: "Apparel",  
    item_category2: "Kids",  
    item_category3: "Shirts",  
    item_category4: "Crew",  
    item_category5: "Long sleeve",  
    price: 20.00,  
    index: 1,  
    item_list_id: "related_products_sidebar",  
    item_list_name: "Related Products",  
    location_id: 'New York Store'  
  }]  
});
```



(default)	ecommerce_attributor	
+ START COLLECTION	+ ADD DOCUMENT	+ START COLLECTION
ecommerce_attributor >	a65b96e6c343aaaca3be98620285cf14b6821485bcc8e82 Hashed client_id	+ ADD FIELD attribution items_data 0 index: '1' item_id: "SKU_123" item_list_id: "related_products_sidebar" item_list_name: "Related Products" location_id: "New York Store" promotion_data: "No promo data" expiration: March 28, 2024 at 9:15:35.124 AM ... id: "a65b96e6c343aaaca3be98620285cf14b6821485...

Example of **select_item** event triggered on website and sent to sGTM endpoint.

Ecom Attributor - Write to Firestore tag is fired, collects necessary data from Event Data Model and stores it in the Firestore.

How to test: Promotion attribution (case No.2)

```
gtag("event", "select_promotion", {  
  items: [{  
    item_id: "SKU_789",  
    item_name: "Google Grey Women's Tee",  
    item_brand: "Google",  
    item_category: "Apparel",  
    item_category2: "Adult",  
    item_category3: "Shirts",  
    item_category4: "Crew",  
    item_category5: "Short sleeve",  
    price: 99.99,  
    location_id: "ChIJIQBpAG2ahYAR",  
    creative_name: 'homepage_banner',  
    creative_slot: 'sidebar',  
    promotion_id: 'P_001',  
    promotion_name: 'Black Friday Sale'  
  }]  
});
```



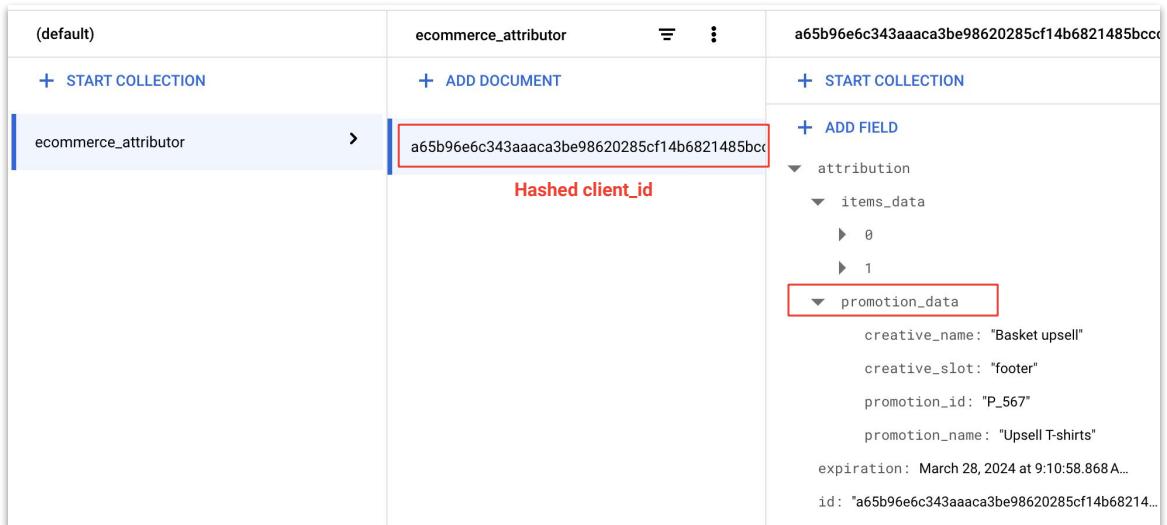
(default)	ecommerce_attributor	a65b96e6c343aaaca3be98620285cf14b6821485bc
<p>+ START COLLECTION</p> <p>ecommerce_attributor ></p>	<p>+ ADD DOCUMENT</p> <p>a65b96e6c343aaaca3be98620285cf14b6821485bcc Hashed client_id</p>	<p>+ START COLLECTION</p> <p>+ ADD FIELD</p> <p>attribution</p> <p>items_data</p> <p>0</p> <p>1 creative_name: "homepage_banner" creative_slot: "sidebar" item_id: "SKU_789" location_id: "ChIJIQBpAG2ahYAR" promotion_id: "P_001" promotion_name: "Black Friday Sale" promotion_data: "No promo data" expiration: March 28, 2024 at 9:05:54.23... id: "a65b96e6c343aaaca3be98620285cf14b6821485bc"</p>

Example of **select_promotion** event triggered on website and sent to sGTM endpoint.

Ecom Attributor - Write to Firestore tag is fired, collects necessary data from Event Data Model and stores it in the Firestore.

How to test: Promotion attribution (case No.3)

```
gtag("event", "select_promotion", {  
  items: [  
    creative_name: 'Basket upsell',  
    creative_slot: 'footer',  
    promotion_id: 'P_567',  
    promotion_name: 'Upsell T-shirts'  
  ]}  
});
```



The screenshot shows the Google Cloud Firestore interface. On the left, under the '(default)' section, there is a '+ START COLLECTION' button and a collection named 'ecommerce_attributor'. On the right, under the 'ecommerce_attributor' section, there is a '+ ADD DOCUMENT' button and a document with the ID 'a65b96e6c343aaaca3be98620285cf14b682148bcc'. This document contains a field 'Hashed client_id' and a nested object 'promotion_data'. The 'promotion_data' object has properties: 'creative_name: "Basket upsell"', 'creative_slot: "footer"', 'promotion_id: "P_567"', 'promotion_name: "Upsell T-shirts"', 'expiration: March 28, 2024 at 9:10:58.868 A...', and 'id: "a65b96e6c343aaaca3be98620285cf14b68214..."'.

In some cases (e.g. if you use legacy Universal Analytics Data Layer implementation), **select_promotion** event doesn't contain item data.

Ecom Attributor - Write to Firestore tag will collect Promotion data and store it inside **promotion_data** object. This data will be sent on event-level with every subsequent ecommerce event, which means if user makes a purchase, Item revenue from all purchased events will be attributed to the last clicked promotion.

How to test: Data sent to GA4

Ecom Attributor - Items array variable is responsible for building new items array with List and Promo information fetched from Firestore.

If we take purchase event as example: purchase event comes into sGTM container. GA4 event tag is fired. Transformation variable called **Ecom Attributor - Augment - Item List data** will replace original items array (which doesn't contain List and Promo info) with **Ecom Attributor - Items array** inside the GA4 tag and information will be sent to GA4 (case no.1 & 2).

In case if there is Promotion information stored in Firestore without the item_id, when GA4 event tag is fired in sGTM container, Transformation variable called **Ecom Attributor - Augment - Promotion data** will add this Promotion information on event-level (case no.3).

Name	Original Value	Final Value
items	[{ item_id: "SKU_123", item_name: "Stan and Friends Tee", item_brand: "You Tube", item_category: "Apparel", item_category2: "Kids", item_category3: "Shirts", item_category4: "Crew", item_category5: "Long sleeve", price: 20 }, { item_id: "SKU_789", item_name: "Google Grey Women's Tee", item_brand: "Google", item_category: "Apparel", item_category2: "Adult", item_category3: "Shirts", item_category4: "Crew", item_category5: "Short sleeve", price: 99.99 }]	[{ item_id: "SKU_123", item_name: "Stan and Friends Tee", item_brand: "You Tube", item_category: "Apparel", item_category2: "Kids", item_category3: "Shirts", item_category4: "Crew", item_category5: "Long sleeve", price: 20, item_list_id: "related_products_sidebar", item_list_name: "Related Products", index: "1", creative_name: undefined, creative_slot: undefined, promotion_id: undefined, promotion_name: undefined, location_id: "New York Store" }, { item_id: "SKU_789", item_name: "Google Grey Women's Tee", item_brand: "Google", item_category: "Apparel", item_category2: "Adult", item_category3: "Shirts", item_category4: "Crew", item_category5: "Short sleeve", price: 99.99 }]

Name	Original Value	Final Value
creative_name		"Basket upsell"
creative_slot		"footer"
promotion_id		"P_567"
promotion_name		"Upsell T-shirts"

List of Tags, Transformations and Variables that are part of JSON file you have to import in sGTM:

Name	Type	Description
Ecom Attributor - Write to Firestore	Tag	Tag used to collect List and Promotion information and write it in the Firestore database
Ecom Attributor - Augment - Item List data	Transformation	Responsible for augmenting items array on ecommerce events which do not contain List information
Ecom Attributor - Augment - Promotion data	Transformation	Responsible for adding Promotion information on ecommerce events if promotion data is not attached to specific item
Ecom Attributor - Items array	Variable	Variable responsible for creating new items array with List/Promotion data from Firestore
Identifier - Event Data - client_id	Variable	Client_id variable from event data model
Identifier - hashed	Variable	Variable for hashing the client_id
Constant - GCP Project ID	Variable	Contains your GCP Project ID. You need to manually paste your GCP Project ID inside this variable
Constant - FS Collection Path	Variable	Contains Firestore Collection Path (by default Collection Path name is: ecommerce_attributor)
FS Lookup - attribution	Variable	Checks Firestore database and returns all List/Promotion data (if it exists for user)
FS Lookup - promotion_id	Variable	Checks Firerstore database and returns promotion_id if it exists
FS Lookup - promotion_name	Variable	Checks Firerstore database and returns promotion_name if it exists
FS Lookup - creative_name	Variable	Checks Firerstore database and returns creative_name if it exists
FS Lookup - creative_slot	Variable	Checks Firerstore database and returns creative_slot if it exists

06

Calculating potential Firestore cost

Firestore cost

Cloud Firestore offers free quota that allows you to get started with your (default) database at no cost. The free quota amounts can be found on this [link](#). Quotas are applied daily and reset around midnight Pacific time. Only the (default) database qualifies for the free quota.

Depending on amount of ecommerce events and users on your website, Ecom Attributor for sGTM solution can produce additional Firestore cost. Below you can find details how to calculate potential cost.

Firestore writes

The cost can be calculated based on events you use as trigger for **Ecom Attributor - Write to Firestore** tag.

For example, if you added following events as triggers:
select_item, select_promotion and purchase events.

In that case, number of writes would be around the same count of select_item, select_promotion and purchase.

Find what is the average daily amount of those events for your website and calculate potential cost using [Pricing table](#).

Firestore TTL policy deletes

Solution uses Firestore TTL policy to automatically delete data.

TTL delete operations count towards document delete cost..

This is bit harder to estimate (because if users return on your website daily, it is possible you will have less TTL deletes than you estimated here), but you can calculate by looking at how many users clicked on select_item or select_promotion events (in 7 day period, you selected to delete data after 7 days).

For delete operations pricing, see [Cloud Firestore pricing table](#).

07

Frequently Asked Questions

Frequently Asked Questions

Does solution support custom ecommerce implementation?

- Solution supports all types of ecommerce measurement implementation (gtag.js, web GTM, third party tag management system, custom implementation). Solution will work as long as ecommerce data received in sGTM respects GA4 ecommerce event data model (event and parameters naming).

Does solution work with advanced Consent Mode?

- Solution doesn't work with unconsented pings. List and Promotion data is stored in Firestore based on hashed client_id, therefore it won't be possible to stitch List and Promotion data from unconsented pings with events after user gave the consent.

Which information is stored in Firestore if I select option "Item List Attribution"?

- Following parameters will be stored in Firestore: `item_list_id`, `item_list_name`, `index`, `location_id`. In case if you have promotion data inside the items array, following parameters will also be collected: `promotion_id`, `promotion_name`, `creative_name`, `creative_slot`. If certain parameters are not set, it simply won't be collected and stored in Firestore.

Which information is stored in Firestore if I select option "Promotion Attribution"?

- Following parameters will be stored in Firestore: `promotion_id`, `promotion_name`, `creative_name`, `creative_slot`. If certain parameters are not set, it simply won't be collected and stored in Firestore.

Will List/Promotion information be collected if it is not located inside items array, if it is actually located inside ecommerce object?

- Yes, solution supports this, information will still be collected. Following logic is applied: solution will always first look into the items array and pick if any information is available. In case if certain parameter is not available inside items array (e.g. `item_list_name`), solution will check inside ecommerce object and pick information if it is available.

Why I need to have Item ID on every ecommerce event that is fired on my website?

- Item ID is required on every ecommerce event (except promotions events) because this is the key how List information is stored in Firestore for every item. If you don't have Item ID on one of your ecommerce events (e.g. item ID missing on begin_checkout event), then List information won't be retrieved from Firestore and sent to GA4.

On which events solution supports collection of List data?

- Solution will support collection of List data (and Promotion data if case if it is available) on following events: `select_item`, `view_item` and `add_to_cart` event. Depending on which event you have List information, those events needs to be provided as triggers to the **Ecom Attributor - Write to Firestore** tag.

On which events solution supports collection of Promotion data?

- Solution will support collection of Promotion data (and List data if case if it is available) only on `select_promotion` event. This event needs to be provided as trigger to the **Ecom Attributor - Write to Firestore** tag.

What if I have only Promotion data on `select_promotion` event (no item data)?

- Solution supports this type of implementation as well. It will collect only Promotion data on `select_promotion` event, and Promotion data will be sent on event-level with any subsequent ecommerce events, which means on purchase event, whole Items revenue will be attributed to the last clicked Promotion.