The Amplitude Guide to Behavioral Data & Event Tracking

Amplitude

In This Guide

03 Preface

04 Key Takeaways

05 What Are Data Types?

- 06 Common data types
- 07 Importance of data types

08 What Is Customer Data?

- 09 Entity data
- 09 Accounts or groups as entities
- 10 Collecting entity data
- 10 Event data
- 11 Collecting event data
- 11 Activating event data

13 What Are the Components of Event Data?

- 14 Naming event data
- 14 Event data components
- 16 Common events and their properties
- 16 Entity types

19 How to Decide Which Events to Track?

- 19 Burning questions
- 20 Events and event properties
- 20 Clicks, views, and processes
- 23 Client-side vs. server-side events

24 How to Create a Tracking Plan?

- 24 Benefits of creating a tracking plan
- 26 The ultimate tracking plan template
- 27 Events
- 29 Event properties
- 31 User properties
- 32 Organization properties

33 Empower Your Team with Robust, Reliable Analytics

Preface

I firmly believe that everybody in an organization, regardless of their role, should understand the basics of data—where it comes from (sources), where it's stored, and where it goes (destinations).

By default, we expect data and engineering folks (collectively referred to as data people) to have a good understanding of these basics as they're the ones tasked with implementing data workflows. But they usually don't have much context in terms of why certain data needs to be tracked and how it will be used in downstream tools.

On the other hand, go-to-market (GTM) folks (or non-data people) know what data they need in their everyday tools but often lack an understanding of where the data comes from and how it's made available to them.

Due to my prior understanding of data integration and enthusiasm toward modern tools and technologies, when I led the customer data infrastructure efforts at a fast-growing SaaS company, I discovered that there are other less-obvious gaps that exist between data people and non-data people.

For instance, extremely proficient software engineers who also doubled up as data engineers were curious to understand why I had them collect behavioral data with such precision and how I intended to use that data beyond analytics.

I found it surprising (and frustrating) that there was a lack of unbiased educational content addressing the questions of both data people and non-data people. I wondered, "If only I could find a resource that explains this really important topic in simple terms, my life and the lives of my engineering counterparts would be much nicer."

Then I told myself, "Maybe I should take some time to create that resource." So I got to work, and the outcome was this guide I originally published as a series of articles under Data-led Academy.

Fast forward, and I've teamed up with Amplitude to accelerate my mission of demystifying data infrastructure for product and growth teams. Amplitude powers data-led insights for hundreds of companies and enables product and growth professionals to own their entire data workflow—from analysis to activation—while also providing necessary tools for them to collaborate with data teams and play an important role in setting up customer data infrastructure.

My hope is that this guide serves as common ground for data and GTM teams and enables them to have better conversations and build better customer experiences, together.

Arpit Choudhury,
Founder, Data Beats and astorik

Arpit is working on bridging the gap between data people and non-data people via the Data Beats community. And at astorik, he's reimagining how SaaS companies build, nurture, and manage relationships with their audiences.

This guide will teach you how to create a tracking plan that accurately captures your behavioral customer data. With the right data foundation in place, you can set your team up for success with analytics they can trust.

Key Takeaways

This guide will help you put your customer data into action and answer the following questions:

- What is customer data and the process of gathering and activating it?
- What do event and entity data look like in the context of customer data?
- Which events should you track and what data should you gather?
- How should you create an event tracking plan?

Once you've answered these questions and created your tracking plan, you can do the following with confidence:

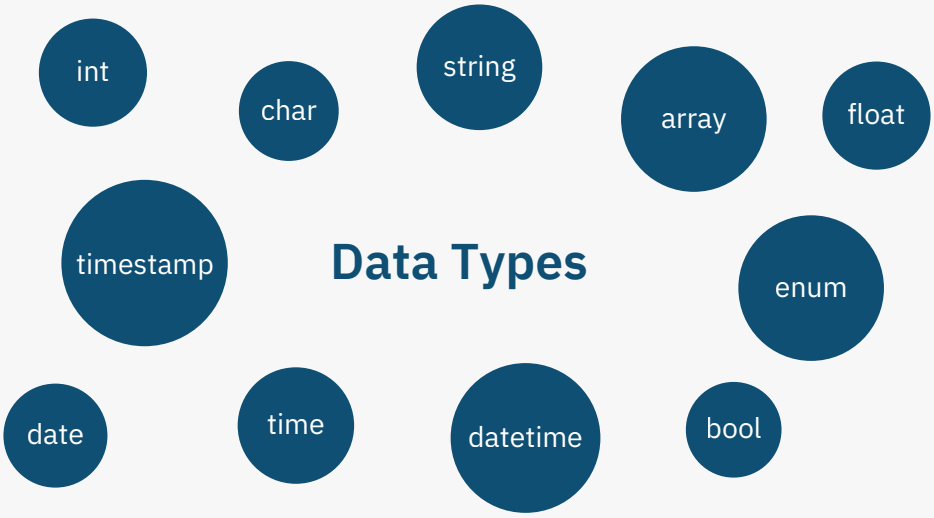
- Lead the implementation of behavioral analytics and engagement tools.
- Gather clean, consistent, and reliable customer data.
- Ask the right questions about your data to better understand user behavior.
- Identify opportunities to collect and activate data to elevate the customer experience.
- Build better products, provide better experiences, and have better conversations.

What Are Data Types?

Before managing your data, you need to understand the common data types you'll come across.

Every piece of data has an attribute, or *type*, that tells a computer system how to interpret that data. For example, knowing the data type for "Ross, Bob" will help a computer understand whether the data is referring to someone's full name ("Bob Ross") or a list of two names ("Bob" and "Ross").

Understanding data types will help you ensure that the data you collect is always in the right format ("Ross, Bob" vs. "Bob Ross") and that the value is as expected ("Ross, Bob" vs. "R0$$, B0b").



Data types should not be confused with the two types of data collectively known as *customer data—entity data* and *event data*. You must define the data type for every entity property and event property to avoid data inaccuracies and data loss.

Common data types

DATA TYPE	DEFINITION	EXAMPLES
Integer (int)	Numeric data type for numbers without fractions	-707, 0, 707
Floating point (float)	Numeric data type for numbers with fractions	707.07, 0.7, 707.00
Character (char)	Single letter, digit, punctuation mark, symbol, or blank space	a, 1, !
String (str or text)	Sequence of characters, digits, or symbols—always treated as text	hello, +1-999-666-3333
Boolean (bool)	True or false values	0 (false), 1 (true)
Enumerated type (enum)	Small set of predefined unique values (elements or enumerators) that can be text-based or numerical	rock (0), jazz (1)
Array	List with a number of elements in a specific order—typically of the same type	rock (0), jazz (1), blues (2), pop (3)
Date	Date in the YYYY-MM-DD format (ISO 8601 syntax)	2021-09-28
Time	Time in the hh:mm:ss format for the time of day, time since an event, or time interval between events	12:00:59
Datetime	Date and time together in the YYYY-MM-DD hh:mm:ss format	2021-09-28 12:00:59
Timestamp	Number of seconds that have elapsed since midnight (00:00:00 UTC), 1st January 1970 (Unix time)	1632855600

Importance of data types

The primary reason to understand each of these data types is to gather clean and consistent data, especially during instrumentation and data collection via surveys. But applying your knowledge of data types is not limited to these activities; data management, data integration, and internal application development (using no-code or low-code tools) also become easier.

Instrumentation

The process of tracking behavioral data from primary data sources and syncing that data to an internal or external storage system is known as instrumentation.

The first step in the instrumentation process is to create a data tracking plan, which is covered later in this guide.

When deciding which events to track and what event and entity properties to collect, specifying the data type of each property in the tracking plan makes the instrumentation process more efficient and leaves little room for error.

This is particularly helpful for engineers who are tasked with instrumentation. You can avoid data inconsistencies by ensuring that each property is sent with the correct data type.

Surveys

As a data-led professional, it's likely that you will gather survey data from your customers at some point during the customer journey.

The questions you ask in a survey could be open-ended (text or number) or come with predefined choices like a drop-down list (enum), checkboxes (array), radio buttons (boolean), or even a slider (depends).

To store the data from surveys in a database or a third-party system, you need to specify a property name (industry_name, job_role, cancellation_reason, is_satisfied, etc.) and its data type (string, number, boolean, etc.) for every field in your survey. The property name stores the value entered, and the data type validates the value is as expected.

If you store data properly and with the right data type, you'll have accurate, consistent, and reliable data that is easier for you to analyze and activate later on.

Keep in mind that open-ended survey questions are harder to analyze because you can't aggregate the responses unless you transform the data by parsing each response and extracting the text that matches a rule. With predefined choices, your analysis is straightforward and unaffected even if you change the choices at a later stage (refer to the enum and array data types).

What Is Customer Data?

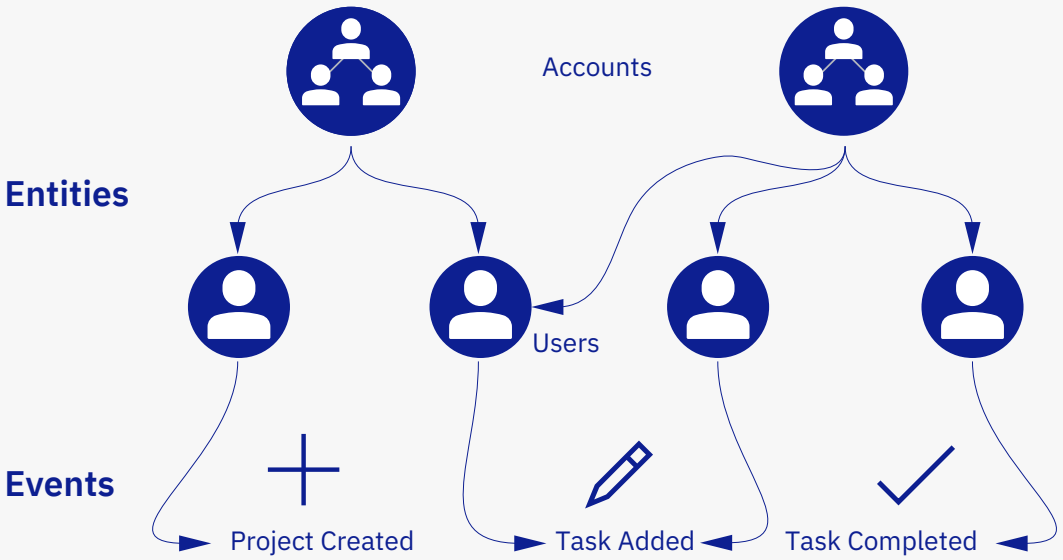
Customer data is the centerpiece that enables personalization and automation at scale—it provides context on the user and their behavior when using a product.

The “customer” in “customer data” includes free users of a paid product and users who pay with personal data to use a product.

Customer data can be broken into the following categories:

- **User data** provides context on a user and their traits. It’s also referred to as *entity data* (the user being the entity).
- **Interaction data** provides context on how the user interacts with a product. It’s also referred to as *event data*, *behavioral data*, or *product-usage data*.

This guide focuses on customer data that comes from a primary or first-party data source—a web app, mobile app, smart device, or combination of these—and comprises entity data and event data. However, customer data is also gathered when users interact with your brand outside of your core product experience via secondary data sources or third-party tools used for advertising, engagement, and support, to name a few.



Entity data

Entity data includes personally identifiable information (PII) such as name, email, and phone number, as well as other details such as age, country, and preferences.

It's often referred to as *user data* since a *user* is the main entity or object. It comprises *user properties* or *user attributes*, each of which stores information or traits about a user.

Entity data is stored in tables where the columns represent user properties like name and email, while each row represents a user. One of the properties acts as an identifier and has to contain a unique value for each row (user).

Name	Email	Country
Jim Morrison	jim@thedoors.com	United States
Zakir Hussain	zakir@musicmaestro.com	India
John Lennon	john@thebeatles.com	United Kingdom

In the table above, *email* can act as an identifier by ensuring no two users have the same email. However, assigning a unique ID to each user is better practice since an email address can change, but the *user_id* remains fixed.

Accounts or groups as entities

A group of users or an account is also an entity with distinct attributes. It's generally referred to as an *organization* or *workspace* for B2B SaaS products.

From a hierarchical standpoint, accounts are groups that comprise users. Thus, the data about an account or group comprises *group properties* that store information about the account, such as the subscription type or the number of users. If accounts are referred to as *organizations*, the associated properties should be called *organization properties*.

It's common to gather data about both users and groups simultaneously. This is particularly true for B2B SaaS tools where a user is part of an account or organization with multiple users.

Collecting entity data

Entity data where *user* is the entity is gathered as a result of users sharing data directly or indirectly.

Users share data directly when they input details in a form, respond to an email or a survey, or interact with conversational interfaces like chatbots and voice bots.

On the other hand, users share data indirectly when they use a product. For example, when a user listens to music on Spotify, they share data about their music preferences, including genres, artists, and even specific songs. Similarly, when a user creates charts in Amplitude, they share data about the type of charts they find useful.

Since Amplitude is a B2B SaaS tool where multiple users are part of an organization, the number of charts created under an organization is data associated with an organization and not a particular user. In this case, *Organization* is another entity, *number_of_reports* is an organization property, and the value of this property changes when any user in an organization creates or deletes a chart.

It's important not to confuse entity data that changes due to product usage *(number of charts)* with event data generated when a user interacts with a product *(chart created)*.

Event data

An event refers to a unique action performed by a user while interacting with a product. The data generated in the process is called event data or interaction data.

Clicks and hovers on the web, taps and swipes on mobile, and text or voice commands on chat and voice interfaces—all of these interactions are actions performed by a user and considered events.

Event data enables you to understand user behavior and is often referred to as behavior data. Additionally, event data enables you to take action on data or activate data in external tools where the data is made available.

A common use case is event-based contextual messaging (in-app or email), where a campaign is triggered when a certain event X takes place. Or when a certain event Y doesn't occur within a specified timeframe after X takes place—the possibilities are endless.

Event data comprises three key elements:

- **The action** or the event that took place
- **The timestamp** or the precise date and time when the event took place
- **The state** or all other properties associated with the event (known as event properties)

Add to Cart, *Buy Now*, and *Complete Payment* are all *actions* or *events*. The exact moment when an event takes place is recorded as a timestamp.

The properties that provide more context about the event *Add to Cart* could be *user_id*, *product_id*, *price*, and *quantity*—all of which provide information related to the event or the state of the event.

Collecting event data

Collecting event data requires you to create a tracking plan specifying the events to track and the associated properties for each event. Then you get your data engineering team to implement the tracking plan using either of the following:

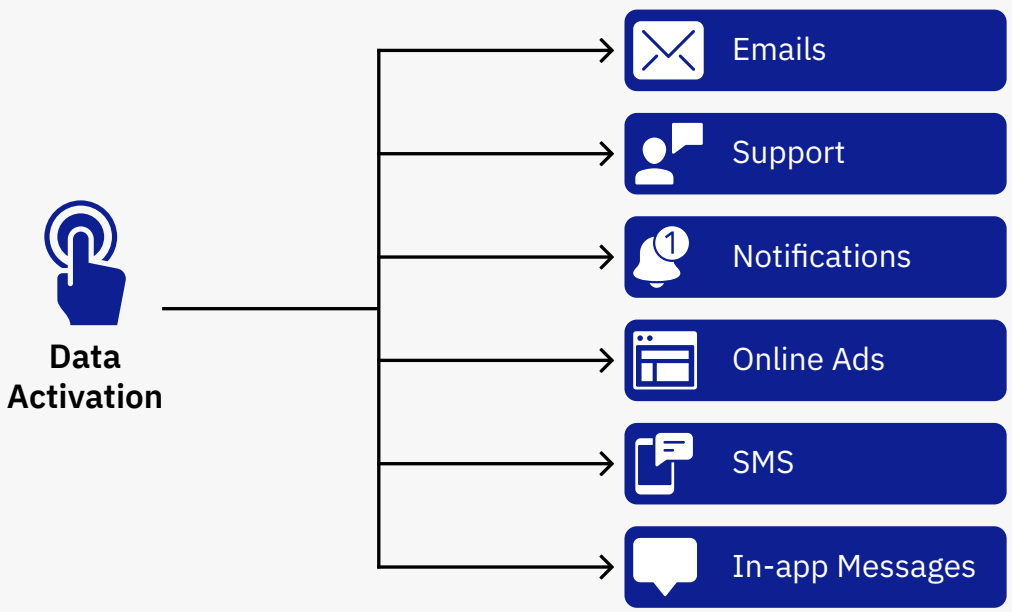
- CDI (customer data infrastructure) or CDP (customer data platform)
- Custom tracking service built in-house

Once event tracking is implemented, event data collected is made available in the configured destinations (product analytics and engagement tools) and typically, a copy of this data is stored in a data warehouse.

Activating event data

Data activation is the process of making data available in the tools used to build data-powered experiences, then building those personalized experiences across customer touchpoints.

In other words, data is activated or acted upon when customer interactions are contextual, timely, and relevant as a result of being powered by data.



Analyzing data and finding insights is only helpful if you can take action on those insights by activating the data in the tools you use to engage with prospects and customers. By activating data efficiently, you can go beyond looking at dashboards and use data effectively.

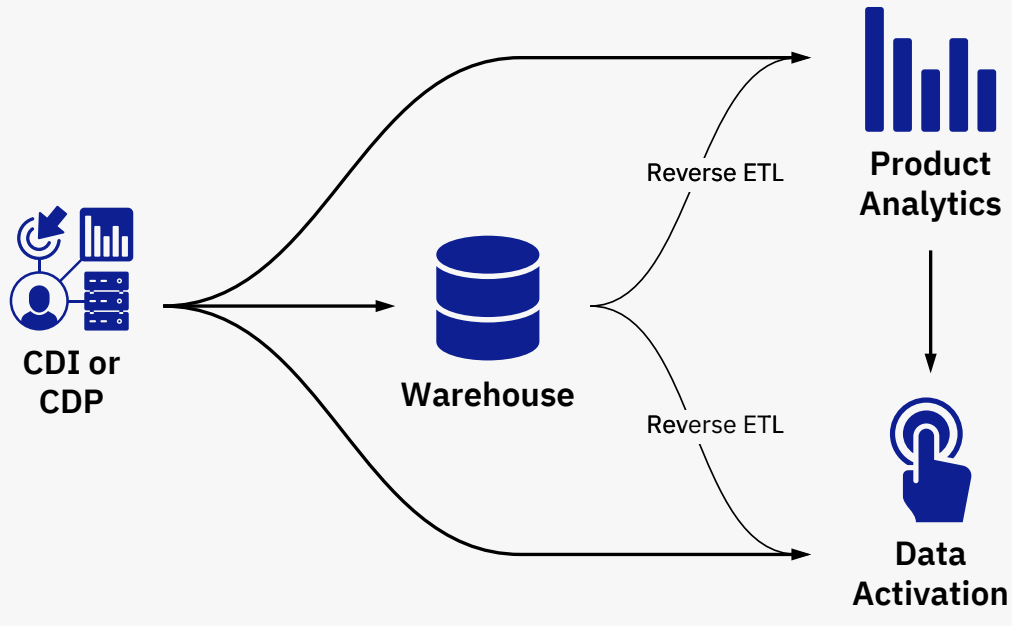
Providing superior customer experiences relies on powering every customer interaction across touchpoints—from outbound emails and support conversations to ads and in-app experiences—with behavioral data. To do so, you need to make data available in the downstream tools where the data is activated.

Without data in those tools, there's not much you can do besides building linear experiences where every customer experiences the same messages, emails, and ads, regardless of the actions they've taken inside your product or the interactions they've had with your brand.

These are the various technologies you can use to move behavioral data to downstream tools where the data is eventually activated. All of these technologies make up the modern data stack:

- **CDIs** to sync raw event data
- **CDPs** to sync raw data as well as visually-modeled data
- **Reverse ETL** to sync data modeled using SQL in a data warehouse
- **Destination integrations by product analytics tools** to sync data already available in those product analytics tools

There are different paths you can take to move data using each of these technologies, but they're all a means to the same end—enabling teams to build personalized, data-powered experiences in the tools they use daily.



The different paths for moving data across your data stack for analytics and activation purposes. Learn more about each of these technologies in our post on the modern data stack.

What Are the Components of Event Data?

Since you have likely bought stuff online, let's start with an ecommerce example.

When interacting with an ecommerce app (web or mobile), you typically buy a product by adding it to your cart, proceeding to checkout, and completing the payment. These are events that you perform when you go through the process of buying an item on the app.

The buyer journey, however, is not so straightforward. Several other events can take place:

- A product is viewed
- The cart is viewed
- A product is removed from the cart
- A coupon is applied
- An address is chosen
- A payment method is chosen
- An order is completed
- And so on

Common events like *Add to Cart*, *Proceed to Checkout*, and *Make Payment* come to mind immediately, but to understand user behavior, you also need to track other events like those mentioned above.

When gathering event data, the first two steps are deciding which events to track and naming the events using a proper naming convention.

Each event is also accompanied by *event properties* (or *event attributes*) that provide more context about the event. Deciding which *properties* to associate with an event and naming those properties are the next two steps in gathering event data.

Steps to gather event data:

1. Decide which events to track.
2. Name those events using a proper naming convention.
3. Decide which properties to associate with each event.
4. Name those properties using a proper naming convention.

Naming event data

What's in a name? When it comes to data, everything.

A proper naming convention or taxonomy distinguishes good data from bad data and enables stakeholders to understand what they are looking at. Not maintaining a standardized taxonomy is one of the main causes for data sets being skewed or bloated with redundancy.

When working with customer data, not maintaining uniform casing when naming events and event properties is one of the biggest mistakes you can make—one that can have long-term ramifications. Strict casing guidelines should always accompany a good naming convention.

Here's why: *Add to Cart*, *added_to_cart*, *productAdded*, *add to cart*, *Added to cart*, and *Product Added* are different ways to define the same event.

While none of these is wrong per se, and there are no set rules for naming events and properties, there are best practices that one should consider following.

The **object-action** naming convention has become the industry standard—and for good reason because it clearly describes the action that has already taken place. *Product Added* definitely means that an object (product) is followed by an action (added).

Learn more about creating a consistent taxonomy for your events with our Fundamentals of Data Taxonomy Design course.

Event data components

There are two key components of an event—an entity (one or more) and event properties.

Associating entity data such as *user_id* with an event like *Signed Up* provides information about the user who performed the event.

Event Data	Entity Data
Signed Up	user_id
Email Verified	first_name
Invite Sent	last_name
Project Created	email
Logged In	job_role
Logged Out	industry_name

Without a unique identifier like *user_id*, event data will remain anonymous, and there will be no way to know who performed the event. Similarly, in B2B SaaS, where a user can potentially be part of multiple organizations, *organization_id* needs to be associated with events to know where events occur.

Besides entities, you can gather additional information for analysis and segmentation when events occur. Returning to the ecommerce example, when a product is purchased, besides knowing *who* made the purchase, you also need to know *what product* was purchased, at *what price*, and *when*. This additional information is gathered in the form of **event properties**.

We mentioned earlier that event data comprises three key elements: the **action** (event), the **timestamp** (when the event occurred), and the **state** (event properties).

Let's look at the event *Product Added* (the name in Proper Case as per the object-action framework for the event *Add to Cart*) and assume that it was performed by a user on *January 1, 2020, at 10 a.m. UTC*. The data gathered when the event took place includes:

- The action: **Product Added**
- The timestamp: **1577872800** (*Unix timestamp for January 1, 2020, 10 a.m. UTC*)
- The state: **0123** (*user_id*), **ABZ** (*product_id*), **7.99** (*price*), and **2** (*quantity*)

In this example, the properties associated with the event *Product Added* are *user_id*, *product_id*, *price*, and *quantity*, each of which provides more information about the event. The *timestamp* is associated with the event to know when it took place.

It's also useful to specify a name for the timestamp, which is essentially an event property. It's not mandatory to do so as the standard practice is to associate the timestamp as *timestamp* with every event when sending data to third-party tools. But specifying a distinct name for the property that stores the timestamp can be helpful in the long run when you need to work with historical event data.

The recommended taxonomy for timestamps is the event name followed by "at"— *product_added_at* for the event *Product Added*.

You might have already noticed that *snake_case* is being used to define event properties, making it easy to distinguish event names from event properties. That said, there are no predefined rules here, and you should choose whatever works best for you and your team.

Here's a final look at the properties associated with the event *Product Added* and the data types of each of those properties:

Event Name	Event Properties	Data Type
Product Added	user_id	String
	product_id	String
	price	Number
	quantity	Number
	product_added_at	Timestamp

Keep in mind that user_id is a user property (entity data) that acts as the identifier for an event and is passed as an event property.

Common events and their properties

Here are some common events and properties that are tracked by most technology products.

Event Name	Event Properties	Data Type
Signed Up	first_name	String
	last_name	String
	email	String
	phone	Number
	country	Enum
	signed_up_at	Unix Timestamp
	user_id	String
Email Verified	email_verified_at	Timestamp
	user_id	String
Signed In	signed_in_at	Unix Timestamp
	user_id	String
Signed Out	signed_out_at	Unix Timestamp
	user_id	String
Invite Sent	invitee_email	String
	invite_sent_at	Unix Timestamp
	user_id	String

Entity types

We previously discussed how entity data relates to event data, but here's a refresher: entity data comprises properties associated with the entity. If **User** is the entity, all information about a user is gathered in the form of user properties. *User_id* is generated for every user by default to identify users and act as an identifier for events.

Types of entity data

Entity data not only helps identify the user (who performs an event) or the organization (under which the event was performed) but also provides additional information about the user and the organization.

It can be helpful to categorize entity data into the following buckets for users:

- **Personally identifiable information** such as *name*, *email*, and *phone*
- **Demographics** such as *age*, *gender*, and *location*
- **Personas** such as *industry*, *job_role*, and *goal*
- **Preferences** such as *brands*, *genres*, and *product_categories*
- **Account data** such as *subscription_type*, *number_of_users*, *account_manager*, and *renewal_date*

The pieces of data under each bucket fall under user properties. User properties store various user details and traits, enabling you to identify and know more about them.

While most of the information comes from the user directly, certain user properties are generated automatically over time due to product usage. But you might wonder, isn't event data also generated due to product usage?

It sure is. User properties are additional details related to an event that are gathered when the event takes place. Let's take a look at the *Signed Up* event and its properties:

Event Name	Event Properties	Data Type
Signed Up	first_name	String
	last_name	String
	email	Number
	phone	Number
	country	Enum
	signed_up_at	Timestamp
	user_id	String

As you can see, all the properties associated with this event provide details about users—details that are either shared by users themselves (*first_name*, *last_name*, *email*, *phone*, *country*) or details that are generated automatically (*signed_up_at*, *user_id*).

Keep in mind the following:

- Some events like *Signed Up* or *Email Verified* are performed only once by every user. The various pieces of data gathered from such events translate into user properties.
- Most user properties except for timestamps and identifiers are subject to change. Users can change their name, email, phone, location, industry, job role, etc. But users can't change the time of signing up (*signed_up_at*) or the unique identifier (*user_id*).

User properties vs. organization properties

With consumer apps, time spent, products purchased, songs played, or videos watched are properties associated with the **user** and stored as user properties, the values of which are constantly updated with an increase in usage.

With B2B SaaS, User and Organization are the main entities, and the events collected are tied to a **user** or an **organization** (or both).

There could be other group entities such as **team** or **project** with certain pieces of data tied to them, as is the case with most productivity tools—the process of gathering **organization** data is also applicable in these cases.

Let's look at common user properties relevant to B2B SaaS products:

User Property Name	Data Type	Expected Values
user_id	String	System-generated ID
first_name	String	User's first name
last_name	String	User's last name
email	Number	User's email address
is_email_verified	Boolean	*True/False*
industry_name	Enum	Predefined enumerators
job_role	Enum	Predefined enumerators
company_size	Enum	Predefined enumerators

When a user is part of an **organization**, many important pieces of information are tied to the organization, not the user. Some common organization properties (also referred to as group properties) are as follows:

Org. Property Name	Data Type	Expected Values
organization_id	String	System-generated ID
organization_name	String	User-specified Name
subscription_name	Enum	*Free, Standard, Business*
subscription_value	Number	Annual Contract Value
is_subscription_paid	Boolean	*True/False*
organization_user_count	Number	Number of users

Keep in mind that the organization_id also acts as an identifier and should be associated with events to know under which organization a certain event took place.

Remember the following statements to help differentiate between user properties and organization properties:

- Every piece of information that helps define **user** cohorts—where they come from, who they are, what their objective is, or what they do inside a product—is stored as a **user property**.
- Every piece of information that helps segment **accounts** or **organizations**—the account type, the revenue it generates, the products or features it uses, the resources it consumes, or the number of users who are part of it—is stored as an **organization property** (or **group property**).

Once you can differentiate between the two, it becomes easy to bring new entities (such as teams or projects) into the mix.

How to Decide Which Events to Track?

To decide which events to track and what data to gather, you need to list questions about your users and their product usage.

Once you start listing your questions, you'll realize that there's so much you want to know. Questions beget more questions, and when that happens, you'll probably want to get all the answers at once. Let's refer to these questions as *burning questions* since that's how this process makes most people feel.

And if you don't feel that way, you might have a strong conviction in your assumptions. But don't let that hold you back from asking questions—you might be pleasantly surprised (or utterly disappointed) when you find out the answers.

It's much easier to ask questions about data when you're able to visualize the data, but this can also be counter-productive if you keep building reports or visualizing data without first asking the burning questions.

Burning questions

Burning questions can be straightforward, like "how many users signed up in the last seven days?" or complex, like "how many users *from the SaaS industry* signed up in the last seven days and *invited another user* to their organization?"

When thinking about burning questions, it helps to start listing down the following actions:

- Actions a user must perform to reach the "aha" moment (activation event)
- Actions that indicate that a user is ready to make a purchase or upgrade an account
- Actions that fuel user engagement and keep a user retained
- Actions that signal that a user is not deriving enough value from the product
- Actions that might lead to user churn

It's also a good time to start questioning the product experience and think through your core offerings. The following questions apply to most technology products:

- What is the time to value, or how long do users take to reach the "aha" moment?
- What are the various paths that users take after signing up?
- What are the points of friction in the user journey?
- What are the most-used features by active users?
- What are the least-used features by paying users?
- What features convert free users into paying users?

Start answering your burning questions today by signing up for **Amplitude's free starter plan**.

Events and event properties

Once you have a list of the burning questions (between five and ten is a good number to start), you can move on to the most critical step—defining events and event properties. This is where you start creating a data tracking plan. Besides the core events, you should also start thinking about the various pieces of data that you want to gather when a particular event takes place. We'll walk through examples of some common events and their associated properties to help you think about this process.

Clicks, views, and processes

It's important to be mindful of the differences between clicks, views, and processes inside your product—every button that is clicked, page that is viewed, or process that is completed can be tracked as a unique event.

In some cases, an event can be tracked as any of the three—a page view, a button click, or a process completion. Let's take a closer look using a hypothetical sign-up flow.

First, the user clicks the sign-up *button* on the homepage to visit the sign-up *page*. Here, the event performed can be tracked as a button click (sign-up button on the homepage) or a page view (sign-up page).

Next, the user fills up the registration form, clicks the submit *button*, and lands on the thank you *page*. If everything goes well, the submission reaches the database, creating a *new row*. Here, the event performed can be tracked as a button click (submit button), a page view (thank you page), or a process completion (new row in the database).

How you choose to track these events depends on your use cases, and sometimes, it might even make sense to track a button click, a page view, or a process completion at the same time.

That said, if your objective is to understand user behavior, you should avoid event redundancy by ensuring that a user action is not tracked multiple times (sign-up *button clicked* and sign-up *page viewed*).

Page viewed

To track page views, you could specify a unique event for each page, such as *Sign Up Viewed*. But that would make your event list too long when you want to track page views for every unique page.

Instead of defining a separate event for each page, you can specify a generic event called *Page Viewed* with these event properties:

Event Name	Event Properties	Data Type	Expected Values
Page Viewed	platform	String	*Web, Android, iOS*
	page_name	String	*Home, Pricing, Sign Up*
	page_variant	String	A or B (if A/B test running)
	utm_source	String	Custom value (facebook)
	utm_medium	String	Custom value (social)
	utm_campaign	String	Custom value (launch)
	page_viewed_at	Timestamp	Unix timestamp
	user_id	String	Unique ID of the user

Button clicked

Like page views, button clicks should also be tracked via a generic event such as *Button Clicked* with these associated properties:

Event Name	Event Properties	Data Type	Expected Values
Button Clicked	platform	String	Web, Android, iOS
	button_name	String	Sign Up, Sign In, Sign Out
	button_location	String	A or B (if A/B test running)
	button_clicked_at	Timestamp	Unix timestamp
	user_id	String	Unique ID of the user

Process completed

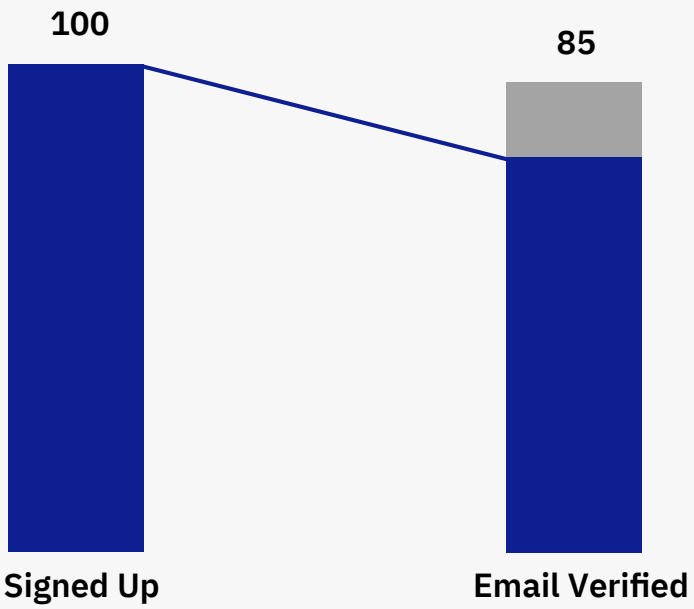
Processes take place as a result of an interaction with a database where data is either written (in a table) or retrieved (from a table). If the interaction fails, then the process fails. Thus, tracking the completion of a process is the most reliable way to track events that rely on completing an interaction with the database.

Here's a common scenario: A user clicks the submit button after filling up the sign-up form only to be presented with a validation error such as "the password must contain a special character." Here, the user performed the event *Button Clicked* but did not actually complete the sign-up process.

Similarly, if the user clicks the submit button, but a server-side error occurs, the process fails, and the user data does not make it to the database. Even though the user submitted the sign-up form successfully, the sign-up process was incomplete. It's crucial to think through the entire process (or the database interaction) that should be completed when an event occurs.

Additionally, you should know if a user signs up for your product but doesn't verify their email address. One way to do this is to check if users log in after signing up (which can only happen after the email is verified), but there could also be users who verify their email and never log in.

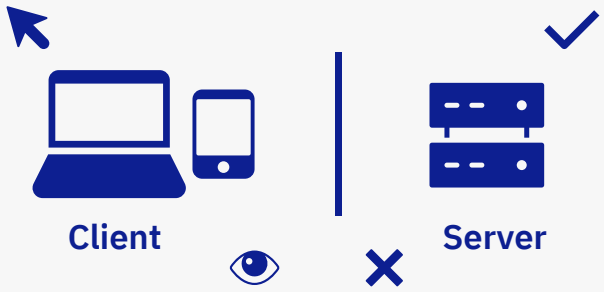
A better approach would be tracking two separate events: *Signed Up* (sign-up process completed) and *Email Verified* (email is verified). This will also tell you how many people sign up but don't verify their email so that you can resend a verification email after a day or two.



Client-side vs. server-side events

Events such as clicks and views that don't rely on database interactions (or backend processes) are essentially client-side events. Client-side events take place exclusively on the client (or the user's device) and are also referred to as frontend events.

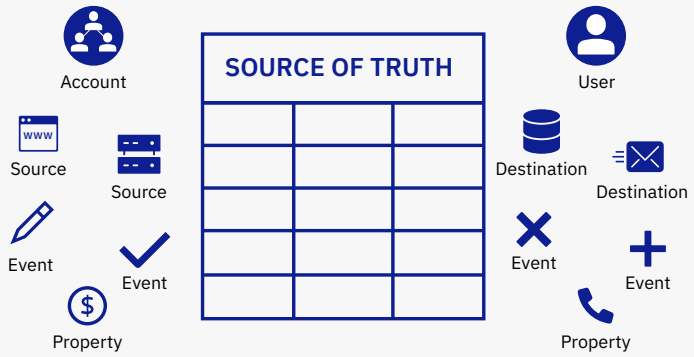
On the other hand, events that rely on backend processes are referred to as server-side events. As the name suggests, server-side events take place on a server when a database interaction is successfully completed. They are also referred to as backend events.



Knowing the difference between client-side and server-side events helps during the instrumentation process because different people in an organization usually implement the two types of events. Even if a full-stack developer is tasked with the implementation of both types of events, it's still helpful to specify the event source in your tracking plan.

How to Create a Tracking Plan?

A *data tracking plan* (usually referred to as a *tracking plan*) is a document that acts as a source of truth for your event data. It's a living document that contains all the information related to the data you gather about your customers when they interact with your product.



Benefits of creating a tracking plan

In practice, a tracking plan is updated every time you wish to:

- Gather a new event, event property, or entity property
- Modify the name of an event
- Modify the name or the data type of a property
- Stop tracking an event or a property

Any piece of data that isn't in your tracking plan shouldn't be tracked, and conducting regular audits of your tracking plan can prevent costly mistakes:

- **Data redundancy**: Gathering the same piece of data multiple times
- **Data inaccuracy**: Gathering inconsistent or incorrect data
- **Data mess**: A combination of redundant and inaccurate data

Besides avoiding these critical issues, there are many benefits of keeping an up-to-date tracking plan.

Ease of implementation

Gathering data is a continuous effort that requires various teams to coordinate and execute new tasks regularly.

A tracking plan not only maintains a repository of what data needs to be tracked but also specifies where the data comes from (data source), the tools and systems the data needs to be sent to (data destinations), and who is responsible for the implementation.

When working with multiple stakeholders, mentioning the purpose of each event or property can expedite the implementation process, especially when one or more people need to approve the tracking of each event or property.

Quick reference

After the implementation, different teams consume and utilize the data across data destinations—the tools and systems where the data is sent. An updated tracking plan enables those teams to know the meaning and purpose of each event or property, making it easy for them to analyze and activate that data.

For example, when a product team wants to conduct a funnel analysis, someone has to create the funnels inside a product analytics tool based on the events made available in the tool.

Similarly, when a growth team wants to run a new email experiment targeting a particular segment of users, they can refer to the tracking plan to easily set up the segment based on available events and user properties in their engagement tool.

Knowledge transfer

Despite its importance, knowledge transfer is usually an afterthought for most companies, especially in the earlier stages of setting up event tracking.

By maintaining a proper and updated tracking plan, companies can avoid many challenges with project handoffs or employee onboarding. For example, enabling new team members to get up to speed with everything that has been implemented for event tracking can save weeks, maybe even months (depending on the amount of data being tracked) of back and forth between new and existing team members.

The ultimate tracking plan template

Now that you know what goes into a tracking plan, let's focus on what it should look like and the best practices to get there.

The tracking plan template below is a Google Sheets spreadsheet created after a lot of research and many iterations to suit almost every industry. It also includes some sample events and properties to give you a head start.

Identifiers	Property Name	Generated When?
User ID	user_id	New user signs up or is invited to join an existing account
Organization ID	organization_id	New or existing *(if applicable)* user signs up for a new account

Done?	Source	Owner	Event Name	Event Description	Property Name
ACTIVATION [Improve User Onboarding and Activation Rate]					
n	Client-side		Signed Up	New user signs up	**user_id** **organization_id** **user_type** **first_name** **last_name** **email** **country** **registration_method** **signup_up_at**
n	Server-side		Project Created	A new project is created	**project_id** **project_name** **project_user_count**

About | Glossary | 1. Events & Event Properties | 2. User Properties | 3. Organization Properties | Discus

Go ahead and **make a copy of the template** (you'll need a Google account to copy the spreadsheet).

This template contains instructions as well as a glossary of terms. Once you review the various tabs, head to the *Events & Event Properties* tab to begin.

We'll now walk through a step-by-step process for creating your tracking plan. The first three steps below apply to all products, while the fourth step only applies to products that need to track account-level activity like B2B SaaS products:

1. Events
2. Event properties
3. User properties
4. Organization properties

Events

Before you proceed, we highly recommend you list the burning questions about your users and their product usage. Ideally, the burning questions should focus on understanding user behavior and the path users take to the activation event.

Additional questions about your product may arise when you start listing events. For example, you might not be sure whether an event should be tracked as a click or a process completion.

Document these questions in your tracking plan under the *Discussions* tab. That way, you won't forget to discuss the most suitable way to define events and properties with your team.

Event Name	Event Properties	Data Type
Signed Up		
Project Created		

List down the events

Under the *Events & Event Properties* tab, list the core events you need to track to answer your burning questions.

The sheet contains sample events *Signed Up* and *Project Created* and their respective properties. For now, ignore the properties and focus on the events.

These two events, for example, can answer the question, "*What percentage of new users created a project?*" Since every event carries a timestamp, you can drill down to fetch data only from a specific period, such as the last seven days.

In this example, *Project Created* is the activation event, and to get to the point where one can perform the activation event, they first have to sign up (perform the *Signed Up* event).

With this example in mind, focus on your activation event and list the key events a user must perform to get to the activation point. Make sure you only list key events that tell you something about the user journey.

By listing too many events, such as every click that leads to the activation event (*Project Created*), you'll end up with too many unnecessary events that will stretch the implementation period and overwhelm you.

Best practice: Start by listing a handful of key events that help understand user behavior and the path to activation—ignore everything else for the time being.

If you still need ideas for what to track, check out Amplitude's implementation guides with suggested taxonomy for ecommerce, fintech, streaming media, print media, and B2B SaaS. Even if your product belongs to another industry, you can gain some inspiration from these or other industry-specific taxonomies.

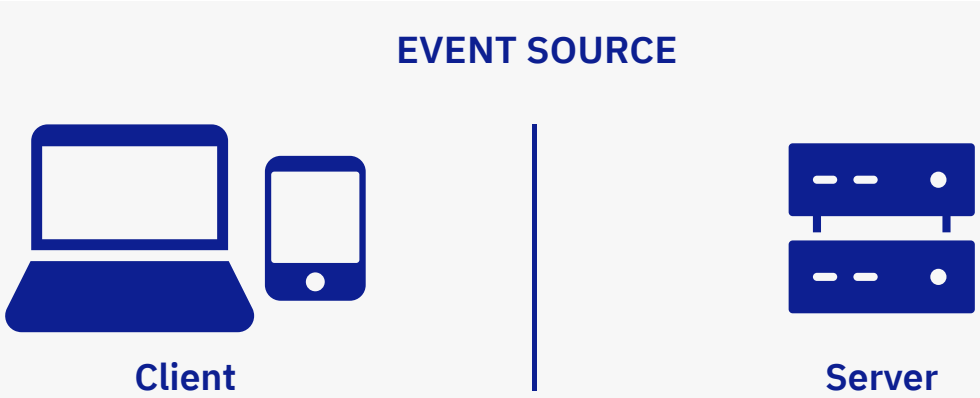
Add the source and owner

It's helpful to mention the source where an event is being tracked. Doing so makes it easy to assign an owner—the person responsible for implementing that event.

In the tracking plan template, *Signed Up* is a client-side event because it takes place on the client as soon as the sign-up form is submitted successfully.

Project Created, on the other hand, is a server-side event that relies on the completion of a process on the server. If the process fails, the event may not be performed even though the user might have completed the steps to create a project.

Whether an event is tracked on the client-side or the server-side depends on your product's architecture and the technologies you use. Specifying it in the tracking plan might require help from your engineering team.



You also might want to track events on external systems connected to your product. For example, if you want to track the event *Support Ticket Created* and use a third-party ticketing tool such as Zendesk, the data source will be Zendesk since that's where the event took place.

Best practice: Bring in data from external sources in the second phase of your instrumentation.

Event properties

Defining event properties after adding events is a better process to follow because you can mentally switch from thinking about events to thinking about the properties that should be associated with every event.

Add the properties for every event

Tackle one event at a time and think about the properties that give you more context about the event.

Keep in mind that a user performs the *Signed Up* event only once (when creating an account), and this is when the unique identifier *user_id* is generated.

If group properties apply to your product, the group identifier *organization_id* is also generated when a user signs up for a new account. If a user is invited to join an existing organization, a new *organization_id* is not generated for that user.

Besides the system-generated identifiers, *Signed Up* carries a property for every piece of information that is gathered from the user when that event takes place (*first_name*, *last_name*, *email*, etc.).

The property *user_type* also helps differentiate those who signed up organically from those who were invited to join an existing account (Invited) or referred by someone to create their account (Referred).

Event Name	Event Properties	Data Type
Signed Up	user_id	
	organization_id	
	user_type	
	first_name	
	last_name	
	email	
	phone	
	country	
	signed_up_at	
Project Created	project_id	
	project_name	
	project_user_count	
	project_created_at	

The event properties associated with *Signed Up* tell you more about the user (PII and traits), so these properties also act as user properties and are added under the *User Properties* tab.

Other events like *Project Created* can be performed multiple times, and the properties associated with the event should only be restricted to those that provide context about one occurrence of that event.

Besides the project ID (*project_id*) and the timestamp (*project_created_at*), the properties *project_name* and *project_user_count* provide additional information every time a project is created (the event *Project Created* is performed).

Additionally, the *user_id* needs to be associated with every event to know who performed the event. If your product needs to track account-level activity, a group identifier such as *organization_id* also needs to be associated with every event.

Specify the data types and expected values

Specifying the data type for every property greatly helps the instrumentation process and is a critical step toward maintaining data consistency.

Event Name	Event Properties	Data Type
Signed Up	user_id	String
	organization_id	String
	user_type	Enum
	first_name	String
	last_name	String
	email	Number
	phone	Number
	country	Enum
	signed_up_at	Unix Timestamp
Project Created	project_id	String
	project_name	String
	project_user_count	Number
	project_created_at	Unix Timestamp

Specifying the expected values for properties is also very helpful for those tasked with the implementation. It enables everyone to be on the same page when a property is supposed to contain predefined values.

Properties with data type *enum* or *array* should always specify the expected values—either the precise values (as done for *user_type*) or a reference to a specific list of values (as done for *country*).

Mention the destinations

The term *destinations* refers to the tools and systems where you wish to send the tracked data. Remember that data should only be sent to destinations where it's consumed or acted upon—not all data should necessarily be sent to all the tools in your suite.

Personally identifiable information (PII) should be handled with utmost care and only sent to tools where PII such as name and email are needed—for example, engagement tools used for event-based messaging (in-app or email).

Best practice: Mention all the destinations for every property, even if most properties are sent to the same destinations.

User properties

User properties store various details and traits about users, enabling you to identify and segment them based on those properties.

As mentioned above, the data gathered from users at the time of signing up are added as user properties because they are user attributes that contain a user's traits and reflect their current state.

Some of these properties remain fixed, while others are subject to change. For example, if a user changes their name on file or the registered email address, the respective properties are updated with the new values. Similarly, the value of the property *is_email_verified* changes to false, and once the new email is verified, *is_email_verified* changes back to true.

User Property Name	Data Type	Expected Values
user_id	String	System-generated ID
first_name	String	User's first name
last_name	String	User's last name
email	Number	User's email address
is_email_verified	Boolean	*True/False*
industry_name	Enum	Predefined enumerators
job_role	Enum	Predefined enumerators
company_size	Enum	Predefined enumerators

Other information gathered from users via surveys, such as the industry they belong to or their job role, is also stored as user properties—they help create user segments for analysis and activation purposes.

Go ahead and list all user properties and their data types, expected values, and destinations in the *User Properties* tab.

Organization properties

Organization properties (or group properties) only apply to products that need to track user activity at an account level.

Like user properties, organization properties store details about organizations or accounts. Any piece of data not associated with a particular user and providing context about the account the user belongs to is stored as an organization property.

Org. Property Name	Data Type	Expected Values
organization_id	String	System-generated ID
organization_name	String	User-specified Name
subscription_name	Enum	*Free, Standard, Business*
subscription_value	Number	Annual Contract Value
is_subscription_paid	Boolean	*True/False*
organization_user_count	Number	Number of users

Group properties are common for B2B SaaS products where a user is part of an account or organization with multiple users. The account name (*organization_name*) and its subscription plan (*subscription_plan_name*) are common group properties that apply to most businesses.

Depending on your product, many more properties could be associated with the account or organization, not a user. Make sure to involve your engineering team and consider whether a property should be a user property or an organization property.

Once you know, list all organization properties and their data types, expected values, and destinations in the *Organization Properties* tab.

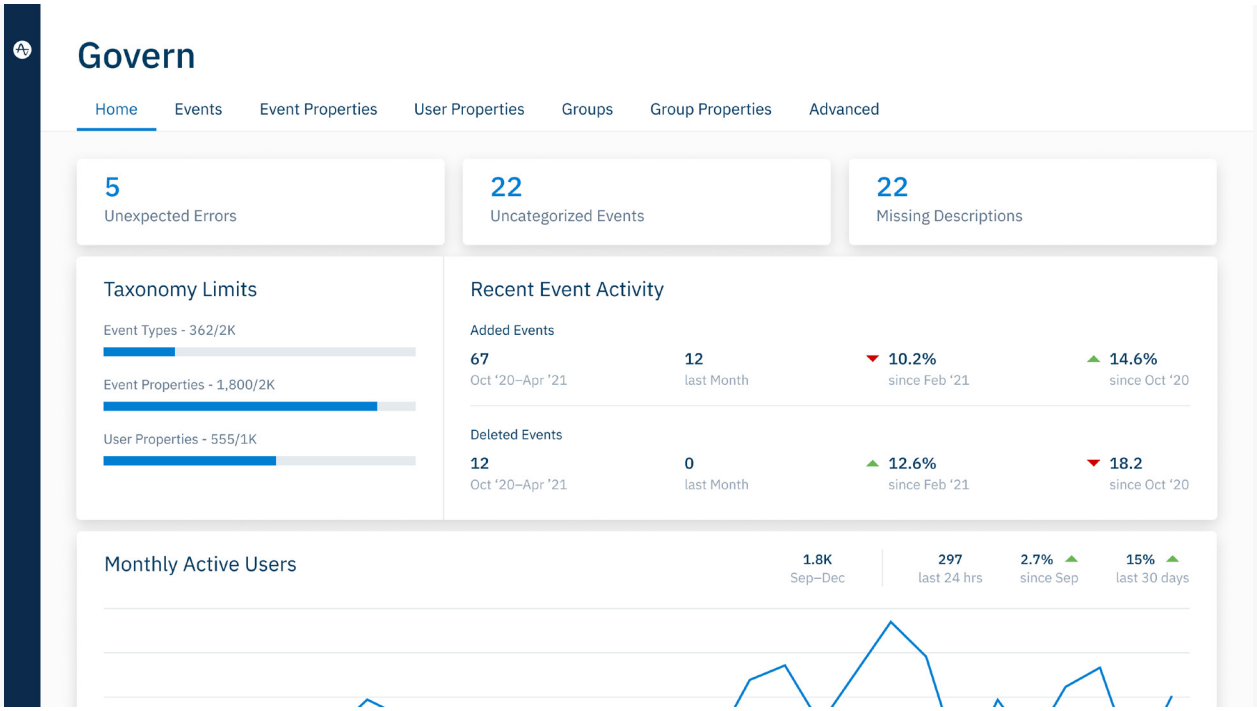
And that's it—the first version of your tracking plan should now be ready to share with your teammates for collaboration. Happy tracking!

Empower Your Team with Robust, Reliable Analytics

Creating an accurate tracking plan is the first step to laying your data foundation and having complete data you can trust. The right data governance, instrumentation, and taxonomy will enable you to modernize your customer data infrastructure.

Once you've set up a data foundation that can scale as you grow, take your customer data to the next level with robust analytics. Lead with confidence knowing that you've done the groundwork and can provide clean, consistent, and reliable analytics across your organization.

And we're here to help. With Amplitude's Data Governance capabilities, you can uplevel event tracking by creating a smart tracking plan to ensure events are correctly instrumented.



Get started for free in Amplitude today.