The Identify call specifies a customer identity that you can reference across the customer's lifetime. There are instances where you want to record information about a user that isn't already known to you. An example of this might be, a user that visits your site and doesn't register, but they do give you their email address through a newsletter email sign-up form. In this instance, you would record that email address as a trait, and for the identifier (ID), you would use anonymous ID.

When you make an identify call using Segment's Analytics.js library, Segment saves the `userId` to the browser cookie, and writes all the user traits in `localStorage`. If you're using one of the Segment mobile libraries, the `userId` and traits are stored in the device's memory. This makes it possible to append the user's data to all subsequent page calls or track calls for the user, so you can properly attribute those actions.

If a user returns to your site after the cookie expires, Analytics.js looks for an old ID in the user's `localStorage`, and if one is found, sets it as the user's ID again in a new cookie. If the user clears their cookies *and* `localStorage`, all of the IDs are removed and the user gets a completely new `anonymousId` when they next visit the page.

Whenever possible, follow the Identify call with a Track event that records what caused the user to be identified.

AnonymousId generation

If you're using Segment's browser or mobile libraries, the Segment SDK generates and sets a UUID as `anonymousID` at the user's first visit to your site. That `anonymousId` is saved in the user's cookie, as well as localStorage, and will stick with that user until the cache is cleared or a `reset` call is triggered.

You can use the `anonymousId` to link events performed by the user as they navigate around your website. When you track the `anonymousId`, you can attribute activities over multiple days to the same user by collecting all of the activities with that ID. If a user chooses to register for your site, or log in to your app, you can Identify them, and still include their `anonymousId` in the event payload along with the new `userId`.

If you use Segment's server libraries, you must generate an `anonymousId` manually. It can be any pseudo-unique identifier, for example, you might use a `sessionId` from a backend server.

Best options for userIds

Segment recommends that you use a unique user identifier (UUID) that won't change for your `userId`. A `userId` should be a robust, static, unique identifier that you recognize a user by in your own database systems. Because these IDs are consistent across a customer's lifetime, you should include a `userId` in Identify calls as often as you can. If you don't have a userId, you need to include an anonymousId in your Identify call in order to record identifying information about your user.

Ideally, the `userId` could be a database ID. For example, if you're using MongoDB it might be a row identifier and look something like `507f191e810c19729de860ea`. These can also be UUIDs that you generate somewhere in your application. You can also use identifiers that you get from other tools - such as Shopify or Braze - however this approach can lead to extra complexity in your systems.

Segment does **not** recommend using simple email addresses or usernames as a User ID, as these can change over time. Segment recommends that you use static IDs instead, so the IDs *never* change. When you use a static ID, you can still recognize the user in your analytics tools, even if the user changes their email address. And even better, you can link your analytics data with your own internal database.

✓
Tip! Even though Segment doesn't recommend using an email address or a username as a User ID, you can still send that identifying information in your Identify call as traits.

When to call Identify

You should make an Identify call in the following situations:

- When the user provides any identifying information (such as a newsletter sign-up with email and name)
- When first you create a user (and so it is assigned a `userId`)
- When a user changes information in their profile
- When a user logs in
- *(Optional)* Upon loading any pages that are accessible by a logged in user

Soft User Registration

An anonymous user visits the site for the very first time. The home page has the analytics.js tracking snippet loaded in its header. When the page loads, this sets off the default Page call to Segment. The Segment SDK generates and sets `anonymousId`.

```
analytics.page({
  path: '/',
  title: 'Home Page',
  url: 'https://somesite.com/',
})
```

You can see in this full page event, the `anonymousId` is populated, and the `userId` is null.

```
{
  "anonymousId": "bd077b70-816b-448b-ae79-2f5f7d856513"
  "context": {
    "ip": "0.0.0.0",
    "library": {
      "name": "analytics.js",
      "version": "3.11.4"
    },
    "locale": "en-US",
    "page":{
      "path":"/"
      "referrer": "",
      "search": "",
      "title": "Home Page",
      "url": "https://somesite.com"
      },
    "userAgent": "Mozilla/5.0"
    },
    "integrations": {},
    "messageId": "ajs-84d32beb4273e661a2257bfef41c4964",
    "originalTimestamp": "2020-04-23T22:38:48.55Z",
    "properties":{
      "path": "/",
      "referrer": "",
      "search": "",
      "title": "Home Page",
      "url": "https://somesite.com"
    },
  "receivedAt": "2020-04-23T22:38:48.55Z",
  "sentAt": "2020-04-23T22:38:48.55Z",
  "timestamp": "2020-04-23T22:38:48.55Z",
  "type": "page",
  "userId": null
}
```

The user signs up for an email newsletter and fills out the form giving you their first and last name, as well as their email address. At this point, you will fire off an Identify call. You won't yet assign them a user ID in this example, but you can still grab these traits about them.

```
analytics.identify({
  firstName: 'Joe',
  lastName: 'Visitor',
  email: 'jvisitor@thissite.com'
});
```

You'll notice the Identify call contains no `userId`. These traits will be associated to the `anonymousId` that is available in the user's cookie and `localStorage`.

```
{
  "anonymousId": "bd077b70-816b-448b-ae79-2f5f7d856513"
  "context": {
    "ip": "0.0.0.0",
    "library": {
      "name": "analytics.js",
      "version": "3.11.4"
    },
    "locale": "en-US",
    "page":{
      "path":"/"
      "referrer": "",
      "search": "",
      "title": "Email Signup",
      "url": "https://somesite.email"
      },
    "userAgent": "Mozilla/5.0"
    },
    "integrations": {},
    "messageId": "ajs-84d32beb4273e661a2257bfef41c4964",
    "originalTimestamp": "2020-04-23T22:38:48.55Z",
    "properties":{
      "path": "/",
      "referrer": "",
      "search": "",
      "title": "Home Page",
      "url": "https://somesite.com"
    },
  "receivedAt": "2020-04-23T22:38:48.55Z",
  "sentAt": "2020-04-23T22:38:48.55Z",
  "timestamp": "2020-04-23T22:38:48.55Z",
  "traits"{
    "email": "jvisitor@thissite.com",
    "first_name": "Joe"
    "last_name": "Visitor"
  },
  "type": "page",
  "userId": null
}
```

Full User Registration

An anonymous visitor registers for an account and becomes a known user. The account creation process allows you to assign a `userId` from your production database and capture additional traits. For this example, the `userId` that is assigned is “123abc”. This is when you’ll want to fire an Identify call with this user’s newly assigned `userId` and additional traits.

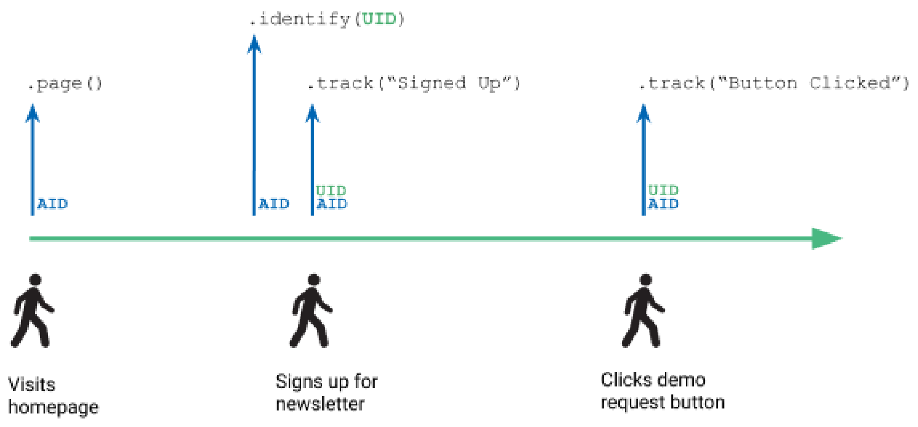
```
analytics.identify(`123abc`,{
  phone: '555-555-5555',
  address: {
    street: '6th Street',
    city: 'San Fransisco',
    state: 'CA',
    postalCode: '94103',
    country: 'US',
  }
});
```

After you fire the Identify call with the `userId`, you’ll notice that the payload now has both a `userId` *and* an `anonymousId` attributed to the user.

```
{
  "anonymousId": "bd077b70-816b-448b-ae79-2f5f7d856513"
  "context": {
    "ip": "0.0.0.0",
    "library": {
      "name": "analytics.js",
      "version": "3.11.4"
    },
    "locale": "en-US",
    "page":{
      "path":"/"
      "referrer": "",
      "search": "",
      "title": "Email Signup",
      "url": "https://somesite.email"
      },
    "userAgent": "Mozilla/5.0"
    },
    "integrations": {},
    "messageId": "ajs-84d32beb4273e661a2257bfef41c4964",
    "originalTimestamp": "2020-04-23T22:38:48.55Z",
    "properties":{
      "path": "/",
      "referrer": "",
      "search": "",
      "title": "Home Page",
      "url": "https://somesite.com"
    },
  "receivedAt": "2020-04-23T22:38:48.55Z",
  "sentAt": "2020-04-23T22:38:48.55Z",
  "timestamp": "2020-04-23T22:38:48.55Z",
  "traits"{
    "phone": '555-555-5555',
    "address": {
    "street": '6th Street',
    "city": 'San Fransisco',
    "state": 'CA',
    "postalCode": '94103',
    "country": 'US',
  }
  },
  "type": "page",
  "userId": "123abc"
}
```

Merging Identified and Anonymous user profiles

The illustration below shows a timeline with a user's interactions on a website, including sample API calls above that show Segment calls, and the user's `anonymousId` and `userId`.



When the user first visits a page, Analytics.js automatically assigns the user an `anonymousId` and saves it to the user's `localStorage`. As the user interacts with the site, for example clicking around to different pages, Analytics.js includes this `anonymousId` and some contextual information with each Page and Track call. The contextual information might be the user's IP address, browser, and more.

When a user signs up to create an account on the website, the `.identify("userId")` and `.track("Signed Up")` events fire, in that order. You pull the `userId` unique to the user from your systems, and send it to the Segment library so you can label that user's later events with their ID. The later Track call ("Signed Up") contains both the `userId` *and* the automatically-collected `anonymousId` for the user, and any other information you capture about them - such as their first name, last name, and email address.

The example below shows an Identify call including user traits. It uses a database ID (`97980cfea0067`) as the `userId`.

```
analytics.identify("97980cfea0067", {
  name: "Peter Gibbons", //user trait
  email: "peter@example.com", //user trait
  plan: "premium" //user trait
});
```

For a Track call, information about this event is stored either in the `context` field or in the event properties. The example below shows a Track call including properties that tell you about the user.

```
analytics.track("Signed Up", {
  userId: "97980cfea0067", //event property
  name: "Peter Gibbons", //event property
  email: "peter@example.com", //event property
  plan: "premium" //event property
});
```

Additionally, Analytics.js adds a `message_id` and four timestamps to the call.

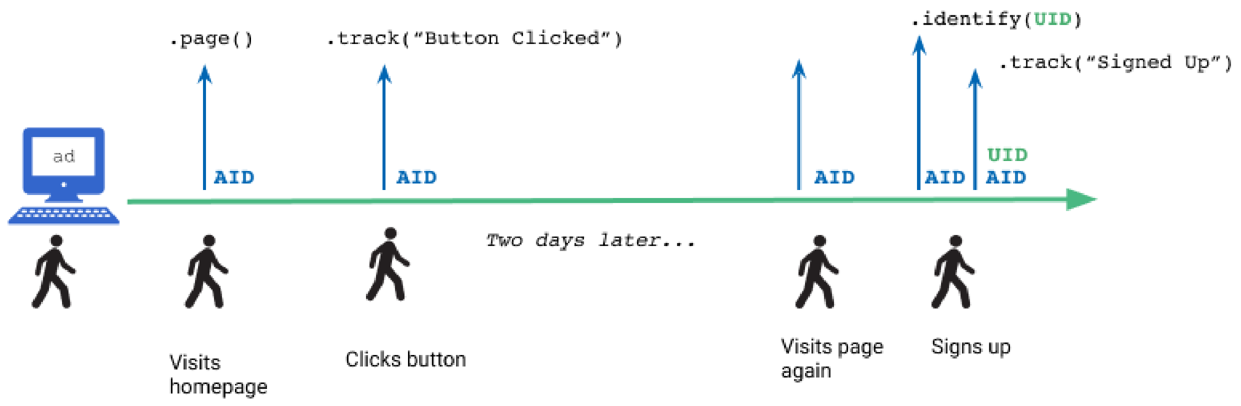
Now, as the user interacts with your site and different buttons or links that you track using Segment, their `userId` and `anonymousId` are sent with each subsequent tracking API call.

UserId merge examples

Let's go through some more scenarios to explain how an `anonymousId` is assigned and how it might be merged with a `userId`.

Scenario #1 - Multi-day, single device

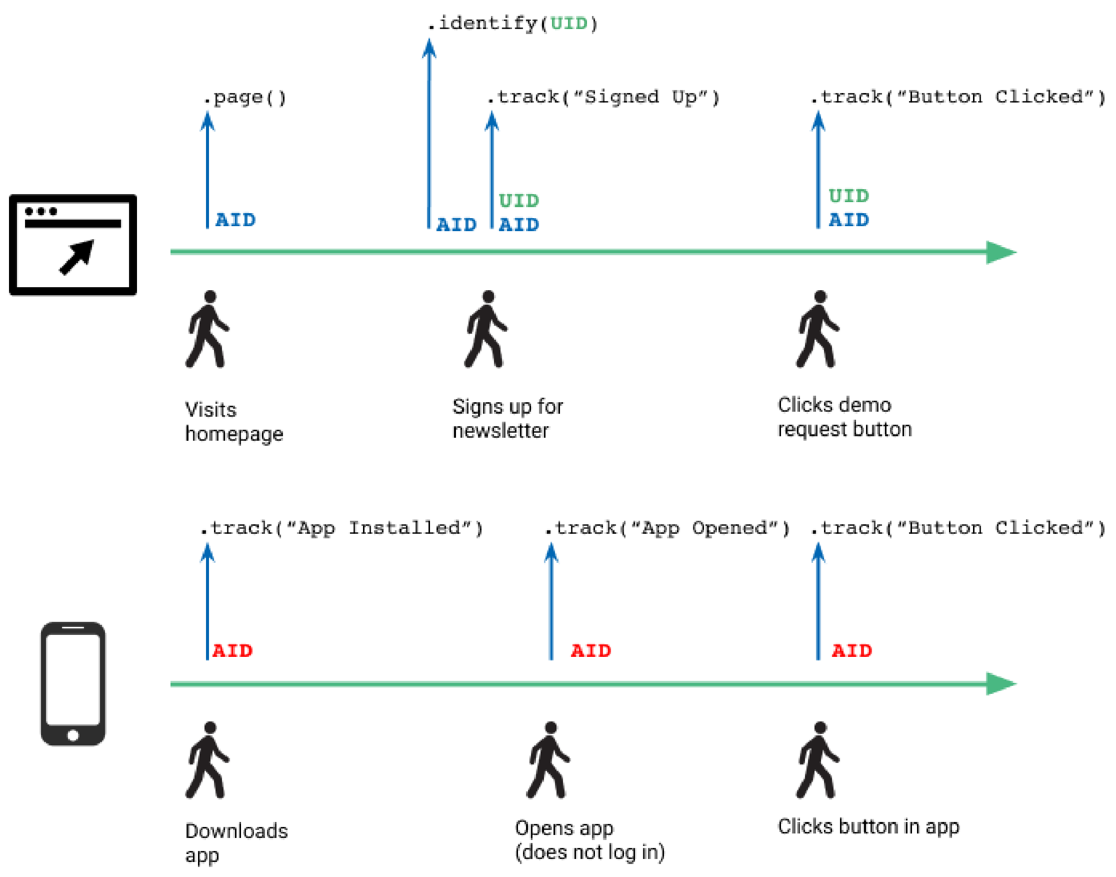
If a user clicks on an ad and is directed to a webpage, they are assigned an `anonymousId`. While this user is anonymous, they navigate to different pages and click around on the website. Say they come back two days later from the same device, sign up, and are assigned a `userId` from your database.



For simplicity, we're assuming that the user has *not* cleared their cookies or `localStorage`, where the original `anonymousId` is stored. If they had, they'd be assigned a new `anonymousId` when they visited the website, and the `userId` they got when they register on the website would *not* be attached to the activities tracked with the old `anonymousId`.

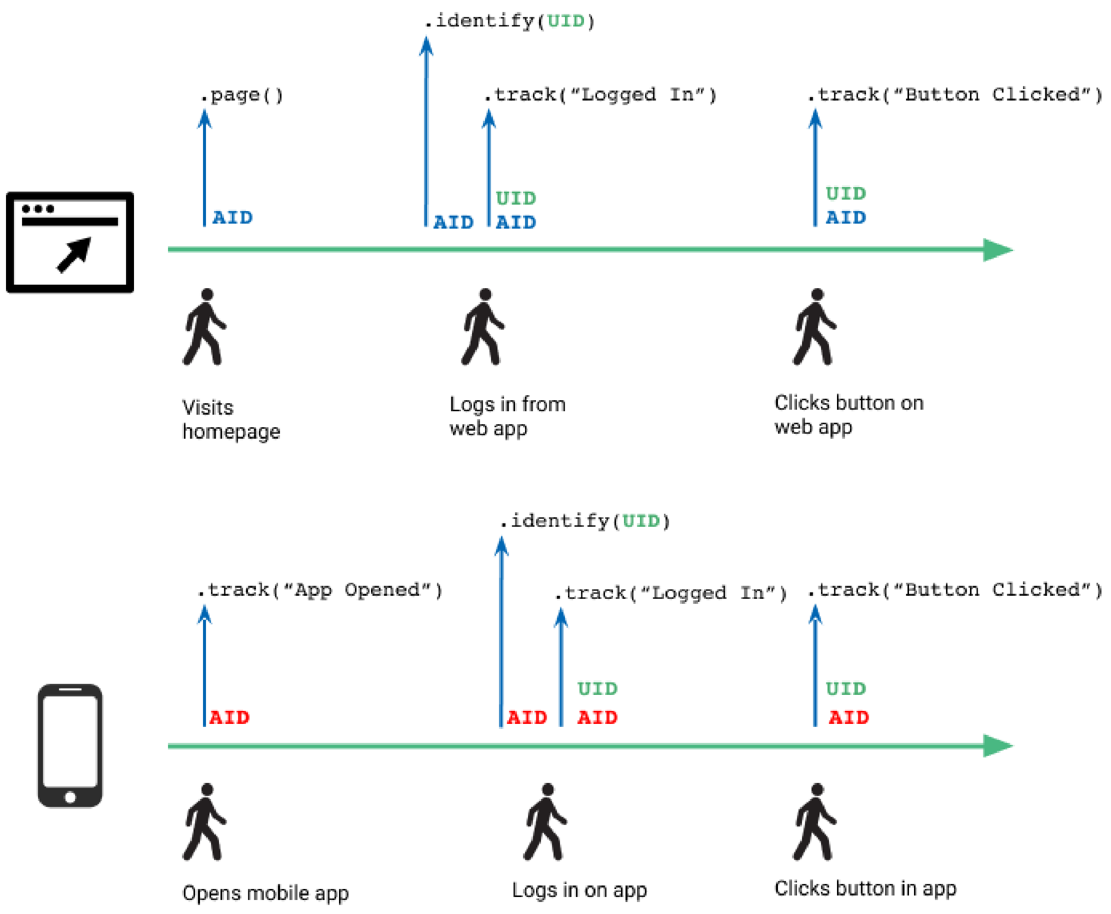
Scenario #2 - Multi-day, multi-device, single login

In this scenario, the person uses both a web browser, and a mobile application to interact with your site. In each case, they are assigned a different `anonymousId`. In this scenario, the user signs up on the web browser, so Segment assigns their *web* session a `userId`. However, because they do not log in on the mobile application, Segment cannot tie the mobile activity to this specific user. Their mobile application activity remains anonymous unless they log in on the mobile application.



Scenario #3 - Multi-day, multi-device, multiple logins

Similar to the previous scenario, the user accessed both your website and mobile application, and also logged in on both. In this case, both sessions on the web and mobile app receive the user's `userId`, so Segment can tie the anonymous activity on both web and mobile to this user.



User profiles in warehouses

Your data warehouse has a schema for each of your Segment sources. User information is stored in two tables in your source schemas - the `identifies` and `users` table.

The `identifies` table contains all of your identify events, and the timestamps for these events. Every time you make an Identify call, Segment adds the `userId`, `anonymousId`, any updated or added user traits from the call, as well as the timestamp of when the call was made. Your `identifies` table is your first stop when you have questions about users and their traits.

The `users` table contains only unique Identify method calls, and is a collation of the `identifies` table. The `users` table is the single source of truth for a user's most up-to-date traits.

These tables only contain information about a user *once they have been identified.* However, you can still find information about an *anonymous* user on the `pages`, `screens`, and `tracks` tables, as well as the individual track event tables.

ID expiration and overwriting

The Segment ID cookie is set with a one year expiration. However, there are some ways an ID can be reset or overwritten:

- If you call `reset` during a user's browser session, it removes both their `userId` and `anonymousId`, which means the user generates a new `anonymousId` on the next visit.
- If the user manually clears their cookies and local storage, they generate a new `anonymousId` on the next visit.
- If you invoke any call before you set an `anonymousId`, Segment automatically sets the `anonymousId` first. This means if you explicitly set an `anonymousId`, you might give the user two `anonymousIds` or overwrite an existing

one.

- If you fetch the `anonymousId` using `analytics.user().anonymousId()` before one is set, Segment generates and sets an `anonymousId` rather than returning `null`.
- If you call `analytics.identify()` with a `userId` that is different from the currently cached `userId`, this can overwrite the existing one and cause attribution problems.
- If you generate a new `anonymousId` on a server library, and pass it from the server to the browser, this could overwrite the user's existing `anonymousId`.

> Remember, if a user has multiple devices, they can have different `anonymousId`s on each different device.

Linking server and client generated Ids

If you're tracking on the client and on the server, the `anonymousId` can be retrieved from `localStorage` on the client and passed to the server. You can access a user's anonymousId using the following call:

```
analytics.user().anonymousId()
```

If you're identifying on the server, then you will want to pass the user ID from the server to the client using an Identify call with the `anonymousId`. That will allow the `userId` to be aliased with the existing `anonymousId` and stored in the cookie in localStorage. With that, all previous anonymous activity and all subsequent activity is associated to the newly generated `userId`, as well as existing `anonymousId`s.

There are some advantages to sending details about your users directly from your server once the user registers. Server library Identify calls are invisible to the end user, making them more secure, and much more reliable. Or, if you want to send user data that is sensitive or which you don't want to expose to the client, then you can make an Identify call from the server with all the traits you know about the user. More about collecting data on the client or server in Segment's documentation.

Aliasing from a server library

If you plan to track anonymous visitors from the browser and only make Identify calls from your server libraries, Kissmetrics and Mixpanel might require that you make an Alias call to link the records. The Alias call links client-side anonymous visitors with server-identified users. This isn't recommended, but if you do this, read the Kissmetrics and Mixpanel specific alias docs.

Common questions

There are a few things that might cause your numbers to be off.

Missing sign-ups

The most common problem people run into when tracking new user signups client-side is that only a portion of their new users are showing up in reports.

This is usually caused by the page redirecting or reloading before the tracking calls get a chance to run. Segment recommends that you make those calls from a welcome page after the user registers, rather than trying to squeeze in the tracking calls on the sign-up page itself.

Anonymous history is lost

This is usually only an issue in Mixpanel, since it's the only destination that requires a call to alias in the browser to link anonymous browsing history to a new identified user.

Remember that for destinations that require aliasing, you must make the Alias call before you make the Identify call for that user. Even if you make an Identify call from a server library, it can't happen before the client-side alias.

Can you update a userId?

Unfortunately, there is no way to change an existing `userId` within Segment. Historical data with an existing `userId` remains the same, and a new `userId` will not replace the existing `userId` in Segment event call logs. For downstream destinations, consult the corresponding docs about user profile behaviors when using a new `userId`.

Changing a `userId` is incredibly hard to do, as that is a fundamental part of analytics. While some downstream analytics tools let you change a `userId` once set, others don't and the process will be different for each tool.

This page was last modified: 15 Mar 2024