Analytics.js sets some default properties when creating cookies for user or group identities. You can override the default cookie properties in code when loading Analytics.js by passing in a `cookie` object to the load method.

> Analytics.js doesn't set third-party cookies and only sets first-party cookies.

Here is the full list of available parameters with their default values:

PARAMETER	DESCRIPTION	DEFAULT VALUE
`domain`	The domain to set the cookie to. This must match the domain of the JavaScript origin. If an Analytics.js cookie already exists at the top-level domain, Segment carries the same cookie value to any subdomains, despite `domain` configuration.	Top-level domain
`maxage`	The maximum amount of time in days before the cookie expires. Browsers may clear cookies before this elapses.	1 year
`path`	The path the cookie is valid for.	`"/"`
`sameSite`	This prevents the browser from sending the cookie along with cross-site requests.	`Lax`
`secure`	This determines whether cookies can only be transmitted over secure protocols such as https.	`false`

Example:

```
analytics.load('writeKey', {
  cookie: {
    domain: 'sub.site.example',
    maxage: 7, // 7 days
    path: '/',
    sameSite: 'Lax',
    secure: true
  }
})
```

To set cookie values using the NPM package, use the following code snippet:

```
    analytics = AnalyticsBrowser.load({
        writeKey: 'writeKey'
    }, {
        cookie: {
          domain: 'sub.site.example',
          maxage: 7, // 7 days
          path: '/',
          sameSite: 'Lax',
          secure: true
        }
    })
```

> Chrome has a maximum limit of 400 days for cookies. If a value is set beyond that, then Chrome sets the upper limit to 400 days instead of rejecting it. Visit Chrome's docs to learn more.

Device-mode destination cookies

Segment doesn't control cookie management for device-mode destinations. As a result, the way cookies are used and managed is solely determined by each individual SDK. For example, if you have concerns about destinations setting third-party cookies, Segment recommends that you consult directly with the destination providers for clarification. For instance, Amplitude, one of the destinations in the Segment catalog, provides an informative article

on this topic.

User settings

Analytics.js automatically persists the user's ID and traits locally. You can override how and where the user ID and traits are stored when loading Analytics.js by passing in a `user` object to the load method.

The user object has the following fields and default values:

OPTION	DESCRIPTION	DEFAULT VALUE
`persist`	This toggles storing user information locally.	`true`
`cookie.key`	Name of the cookie used to store the user ID.	`ajs_user_id`
`cookie.oldKey`	Name of a cookie previously used to store the user ID. Will be read if `cookie.key` can't be found.	`ajs_user`
`localStorage.key`	Name of the key used to store user traits in localStorage.	`ajs_user_traits`

Example:

```
analytics.load('writeKey', {
  user: {
    persist: true,
    cookie: {
      key: 'ajs_user_id'
    },
    localStorage: {
      key: 'ajs_user_traits'
    }
  }
})
```

Group settings

Analytics.js automatically persists the user's group ID and group properties locally. You can override how and where the group ID and properties are stored when loading Analytics.js by passing in a `group` object to the load method.

The group object has the following fields and default values:

FIELD	DESCRIPTION	DEFAULT VALUE
`persist`	Toggles storing group information locally.	`true`
`cookie.key`	Name of the cookie used to store the group id.	`ajs_group_id`
`localStorage.key`	Name of the key used to store user traits in localStorage.	`ajs_group_properties`

Example:

```
analytics.load('writeKey', {
  group: {
    persist: true,
    cookie: {
      key: 'ajs_group_id'
    },
    localStorage: {
      key: 'ajs_group_properties'
    }
  }
})
```

Persistent retries

When enabled, Analytics.js automatically retries network and server errors. When the client is offline or your application can't connect to Segment's API, Analytics.js stores events in `localStorage` and falls back to in-memory storage when `localStorage` is unavailable.

Disable all client-side persistence

Analytics.js supports disabling persisting any data locally. This will force analytics.js to store data in-memory only and disables automatic identity tracking across pages.

You can completely disable client-side persistence when loading Analytics.js by setting `disableClientPersistence` to `true`.

```
analytics.load('writeKey', { disableClientPersistence: true })
```

Identity

When `disableClientPersistence` is set to `true`, Analytics.js won't be able to automatically keep track of a user's identity when navigating to different pages. This can cause increased MTU usage if the anonymous usage can't be associated with a `userId`.

You can still manually track identity by calling `analytics.identify()` with the known identity on each page load, or you can pass in identity information to each page using the querystring API.

Event retries

Analytics.js tries to detect when a page is about to be closed and saves pending events to `localStorage`. When the user navigates to another page within the same domain, Analytics.js attempts to send any events it finds in localStorage.

When `disableClientPersistence` is set to `true`, Analytics.js won't store any pending events into `localStorage`.

Client-side cookie methods (get, set, clear)

To access or assign a value to a cookie outside of the standard Segment methods (track/identify/page/group), you can use the following methods. To access the cookie's value, pass an empty `()` at the end of the method. To assign the value, include the string value inside those parenthesis, for example, `('123-abc')`. To clear or remove the value for a specific field, pass in an empty value of its type. For example, for string `('')`, or for object `({})`.

FIELD	COOKIE NAME	ANALYTICS.JS GET METHOD	LOCAL STORAGE GET METHOD	SET EXAMPLE	CLEAR EXAMPLE

userId	ajs_user_id	analytics.user().id();	window.localStorage.ajs_user_id	analytics.user().id('123-abc');	analytics.user
anonymousId	ajs_anonymous_id	analytics.user().anonymousId();	window.localStorage.ajs_anonymous_id	analytics.user().anonymousId('333-abc-456-dfg');	analytics.user
user traits	ajs_user_traits	analytics.user().traits();	window.localStorage.ajs_user_traits	analytics.user().traits({firstName:'Jane'});	analytics.user
groupId	ajs_group_id	analytics.group().id();	window.localStorage.ajs_group_id	analytics.group().id('777-qwe-098');	analytics.grou
group traits	ajs_group_properties	analytics.group().traits()	window.localStorage.ajs_group_properties	analytics.group().traits({name:'Segment'})	analytics.grou

To retrieve a specific user trait using the Analytics.js Get method, you can access the trait by invoking `analytics.user().traits().firstName`. This returns the firstName trait of the user.

To retrieve a specific group trait, you can use the method `analytics.group().traits().companyName`. This returns the companyName trait of the group.

When you access specific traits stored in the browser's localStorage, you need to utilize the JSON.parse() method because the stored data is typically in string format.

Storage Priority

By default, Analytics.js uses `localStorage` as its preferred storage location, with Cookies as a fallback when `localStorage` is not available or not populated. An in-memory storage is used as a last fallback if all the previous ones are disabled.

Default Storage Priority:

```
LocalStorage -> Cookie -> InMemory
```

Some scenarios might require a switch in the storage systems priority:

- Apps that move the user across different subdomains
- Apps where the server needs control over the user data
- User Consent
- Availability

You can configure the storage priority in the Analytics.js client using the `storage` property, either globally or only for user or group data.

The `storage` property accepts an array of supported storage names (`localStorage`, `cookie`, `memory`) to be used in the priority order of the array.

```
analytics.load('writeKey', {
  // Global Storage Priority: Both User and Group data
  storage: {
    stores: ['cookie', 'localStorage', 'memory']
  },
  // Specific Storage Priority
  user: {
    storage: {
      stores: ['cookie', 'localStorage', 'memory']
    }
  },
  group: {
    storage: {
      stores: ['cookie', 'localStorage', 'memory']
    }
  },
}
```

This page was last modified: 18 Dec 2024