

User attributes

Last Modified on 19.01.2026

About attributes

Attributes are fields that you can add as optional to your user profiles. They can be filled with data.

In this article we cover

- the benefits
- when to use it
- the implementation via SSO
 - reviewing what data is passed on
 - update the attributes
- the implementation within the Snippet
 - reviewing what data is passed on
 - update the attributes
- the implementation via CSV file upload (Browser Extension only)

Why use it

- **Personalization:** Custom data allows you to tailor user experiences by capturing specific user attributes relevant to your application(s) or the company as a whole.
- **Targeting:** With custom attributes, you can create user segments based on specific criteria, enabling targeted messaging, content, or features for different user groups.
- **Analytics:** Custom attributes help gather valuable data for analysis, allowing you to gain insights into user behavior, preferences, and trends.
- **Automation:** By leveraging custom attributes, you can automate processes or workflows based on user attributes, such as triggering personalized Guides, tooltips or announcements.
- **Flexibility:** Custom attributes provide flexibility to adapt and extend user profiles as per your application's evolving needs, ensuring scalability and customization options.

When to use it

If you want to pass on data to target specific user groups. You can use the attributes to base your User Segments on them.

Before you proceed, make sure you've already planned and created all custom data fields that you will need.

The Implementation via SSO

When deploying Userlane using **SSO (Single Sign-On)**, it is critical to ensure that all **required custom user attributes** (e.g., `role`, `department`, `region`, `language`) are **transmitted via the SAML assertion**.

NOTE: You can create those custom attributes in the Portal in advance & map them while setting up SSO. Otherwise, custom attributes will automatically be created in Portal while transmitting data from IdP the way they presented in there

You can learn more about this [here](#).

Check what Attributes are passed on

Once SSO succeeds, validate that your attributes (`region`, `role`, etc.) are correctly passed and available.

Go to **Userlane Portal > Settings > Company Users**.

Click on a Company User and review what information is passed. If any attribute is missing, please have add it following the instructions [here](#).

Update the Attributes

Updates are tied to SSO authentication events.

Whenever a user logs in via SSO, Userlane fetches the latest attributes from the IdP and updates the profile.

To control frequency, you configure the SSO Refresh Interval in the Portal:

- Disabled (no refresh)
- Daily (24h)
- Every other day (48h)
- Weekly (7 days)

This interval determines when already-authenticated users will have their attributes re-synced.

The Implementation within the Snippet

1. Your general Snippet

Every account has a unique account ID which is also included for the Java Script snippet to load Userlane within an application. This can be found in the Userlane Portal.

2. Know where to amend the Snippet

To amend the snippet to save data in the user profile of the current user, you need to add an identify command

in the snippet:

```
Userlane('identify', currentUser.id, {  
  attribute_key: 'attribute_value'  
});
```

Example snippet:

```
// load Userlane  
  
(function(i,s,o,g,r,a,m){i['UserlaneCommandObject']=r;i[r]=i[r]||function(){(i[r].q=i[r].q||[]).push(arguments)};a=s.createElement(o),m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)})(window,document,'script','https://cdn.userlane.com/userlane.js','Userlane');  
  
  
// TODO: the command below is just for illustration purposes  
// it's definition must be changed to work for your application  
  
var currentUser = please_replace_this_with_your_api_call.getCurrentUser(); // returns an object with id and other data  
of the current user who's using the application in this moment  
  
  
// identify the user for Userlane  
  
Userlane('identify', currentUser.id, {  
  
  // TO DO: these fields below are defined by you.  
  // Below are just examples. You must individually customize your own attributes and variables here.  
  
  name: currentUser.name,  
  email: currentUser.email,  
  permissions: currentUser.permissions  
});  
  
  
// initialize Userlane with your account ID, in this example it is "12345"  
Userlane('init', 12345);
```

3. Decide which custom field data type to use

Every custom field has a data type. It can only accept data that fits this type. These types are defined when the custom data field is first created and can not be changed afterward.

Supported data types:

- **True/false:**

Custom true/false fields accept JavaScript booleans. We also parse values like 1, 0, “yes” and “no” to their respective true/false value.

Example:

```
Userlane('identify', currentUser.id, {  
  is_admin: true  
});
```

- **Text:**

Custom text fields accept JavaScript strings. There is no length limit on individual strings but all custom data may not exceed the size of 500kb.

Example:

```
Userlane('identify', currentUser.id, {  
  description: 'Some text description'  
});
```

- **Number:**

Custom number fields accept JavaScript numbers, both integers and decimals/floats.

Example:

```
Userlane('identify', currentUser.id, {  
  store_number: 224  
});
```

- **Date and time:**

Custom date and time fields accept strings that represent a timestamp in any of these formats:

● YYYY-MM-DD HH:mm:ss	● 2017-04-28 20:12:55
● yyyy-mm-ddThh:mm:ssZ	● 2017-04-28T20:12:55Z
● yyyy/mm/ddThh:mm:ssZ	● 2017/04/28T20:12:55Z
● yyyy-mm-dd	● 2020-02-20
● yyyy-mm-dd hh:mm:ss	● 2020-02-20 23:42:51
● yyyy/mm/dd hh:mm:ss	● 2020/02/20 23:42:51
● yyyy/mm/dd	● 2020/02/20

Example:

```
Userlane('identify', currentUser.id, {  
  date_created: '2020/02/20 23:42:51'  
});
```

- **Array (List):**

Custom list fields accept JavaScript arrays. We only accept string and integer elements in the array. The order of the elements in the array is not considered. When a new array is provided, we'll override the previous array entirely.

Example:

```
Userlane('identify', currentUser.id, {  
  roles: ['admin','editor','user']  
});
```

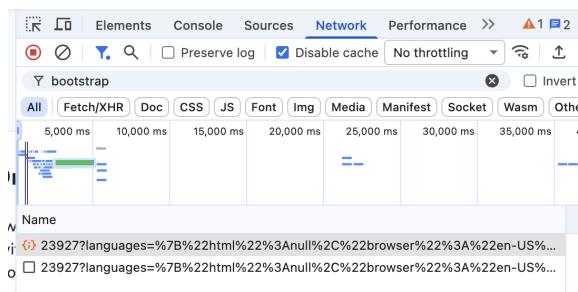
i An Attribute that is passed via the snippet but has not been set up in the Portal will be ignored by Userlane. This also applies to Attributes that have a different name in the snippet and in the Portal.

After creating the Attributes, you can create a [User Segment](#) you can then apply to specific Userlane content.

Check what Attributes are passed on

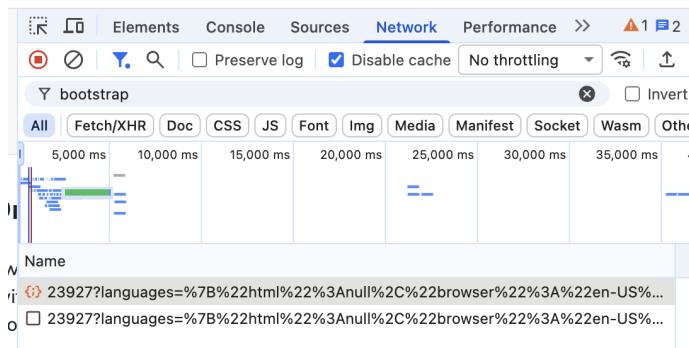
When you already implemented the Snippet, you can check in the Developer Tools what information is being passed on.

1. Open the Developer Tools and navigate to the Network tab
2. Once in the Network tab, enter 'bootstrap' in the filter:

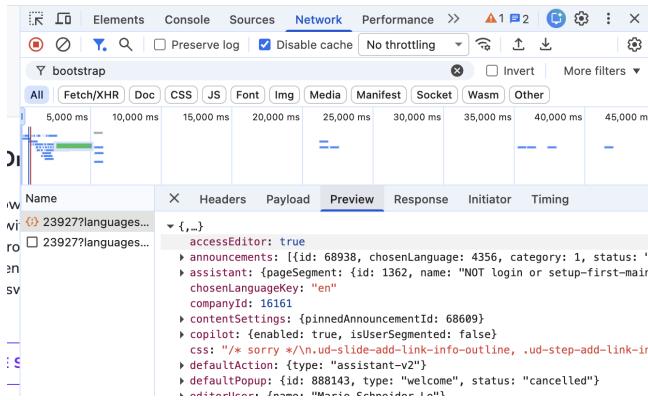


3. After entering 'bootstrap', refresh the page to re-trigger the Userlane commands

4. Then you will find two languages options under the 'Name' column, click to open



5. Upon opening, you will be able to see the implemented attribute keys as well as their values for your users in the 'Payload' tab under customerUser/customData



Update the attributes

With the Snippet, attributes update whenever your application sends a new `identify` call, so it is entirely dependent on what information is being passed on via the Snippet.

The implementation via CSV file upload (Browser Extension only)

If you want to upload the CSV file with the list of your users and their data you can pass user data such as: `region` , `role` , `department` , `jobtitle` , etc. as custom user attributes.

In order to start you should upload CSV file with your users and map users data from the file onto the custom attributes available in Portal. More info on how to upload CSV file you can find [here](#).

NOTE: you can add missing custom attributes right while mapping data from CSV file onto Portal attributes or add custom attributes in advance

If you want to update any attribute, you have to upload an updated CSV file.