Custom JavaScript Transformations on Common Resources

Last Modified on 08/31/2021 9:37 am EDT

Custom JavaScript can be used to manipulate fields from a transformation object to fit the structure and format of a common resource in ways that cannot be achieved through mapping. Suppose that your transformation's contact object has the individual fields `firstName` and `lastName` yet your common resource uses a single field, `fullName`. Custom JavaScript can be used to combine the values for `firstName` and `lastName` to populate your common resource's `fullName` field with a singular value for the contact's name.

How to Add Custom JavaScript

> **Note:** This article assumes that you already have created a common resource and at least one transformation for that resource. For more information on, see Creating common resources.

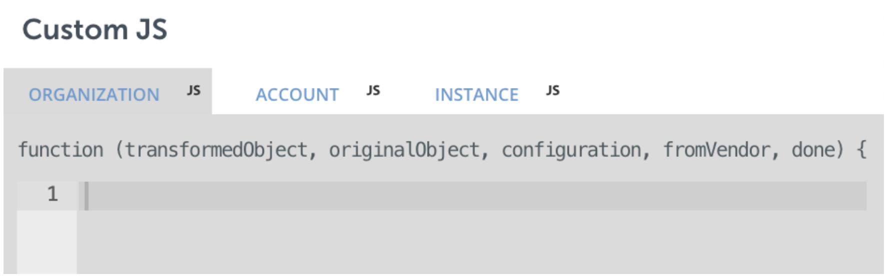
To get to a common resource:

1. Access the page.
2. Select a resource from the My Resources list.
3. Open one of your transformations from the Mapped Transformations pane on the right side of the screen.
4. Click in the upper-right corner to open a code editor where you can insert your custom JavaScript.

Tips for Custom JavaScript

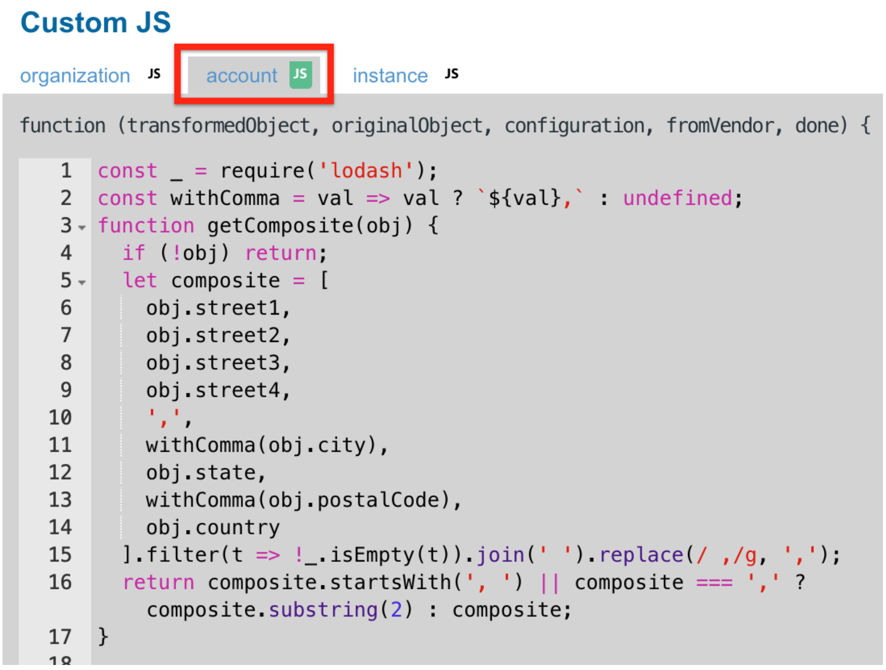
When using custom JavaScript to transform common resources, you enter the script on the tab indicating the level at which you want the it to execute in the transformation. Following the Account -> Instance hierarchy, Valid JavaScript code entered at higher levels will be inherited by lower levels, but only if the lower levels do not contain JavaScript code of their own.

For example, if you enter code at only the account level, the account-level code will be executed at the account and instance levels. However, if you enter code at both the account and instance level, the account-level code will be executed at the account and instance levels, but only the instance-level code will be executed at the instance level.



```
function (transformedObject, originalObject, configuration, fromVendor, done) {
```

A green JS icon indicates that something has been entered on the respective tab; however, note that a green JS icon only shows that content has been added to the tab, not necessarily that the content is validated JavaScript.



```
function (transformedObject, originalObject, configuration, fromVendor, done) {
const _ = require('lodash');
const withComma = val => val ? `${val},` : undefined;
function getComposite(obj) {
  if (!obj) return;
  let composite = [
    obj.street1,
    obj.street2,
    obj.street3,
    obj.street4,
    ',',
    withComma(obj.city),
    obj.state,
    withComma(obj.postalCode),
    obj.country
  ].filter(t => !_.isEmpty(t)).join(' ').replace(/ ,/g, ',');
  return composite.startsWith(', ') || composite === ',' ?
    composite.substring(2) : composite;
}
```

Common JavaScript Transformations

Below are some code samples for common JavaScript transformations. You can find more examples in the Examples section of Transforming Fields. In all of these code samples, data from the `originalObject` argument is used to create new fields on or otherwise modify the `transformedObject` argument. When you have finished modifying/creating your `transformedObject`, you will need to call `done(transformedObject);` at the end of your JavaScript.

To test the results of your JavaScript, you can use click the play icon button which will show you the request being made along with the original and transformed object. You can also test the transformation using a specific object by using the ID field and rerunning the test.

Additionally, it is mandatory at the end of custom JavaScript to call

```
done()
```

if code and/or comment has been added in order for the code/comment to be parsed correctly.

Combining Fields

```
transformedObject.fullName = `${originalObject.firstName} ${originalObject.lastName}`;
```

Dividing Fields

```
let splitName = originalObject.fullName.split(' ');

 transformedObject.firstName = splitName[0];

 transformedObject.lastName = splitName[1];
```

Replacing Unwanted Characters in Fields

```
transformedObject.fieldWithoutNewLines = originalObject.field.replace(/\r?\n|\r/g, "");
```