Logging Data

During API Extensions development, it is helpful to use the available logging tools to examine the data your functions are interacting with. The process is simple: expose the places in your code where you want to log data about a particular object or result, and then view the logged data through the Action Logs available in Dev Center. After you are done with development, make sure to go back into your code to remove or minimize the amount of logging from your application to ward off performance issues.

Log Data During Development

To log data during development and debugging, use the console module available in Node.js and configure the Action Management JSON Editor to allow low priority logs to display in Dev Center:

1. Open an action file in the `assets/src` directory and log data to the console using `console.log` or `console.info`. A great use case for logging is to view what data the `context` object provides different actions, as shown in the following two examples (which include an embedded action and an http action), but you can log just about any data you want.

```
module.exports = function(context, callback) {
 console.info(context);
 console.info(context.get.cart());
 console.info(context.get.cartItem());
 console.info(context.apiContext);

 // Rest of code

 callback();
};
```

`assets/src/domains/commerce.catalog.storefront.shipping/`

`http.commerce.catalog.storefront.shipping.requestRates.after.js`

```
module.exports = function(context, callback) {
 console.info(context);
 console.info(context.request.headers);
 console.info(context.request.params);
 console.info(context.request.href);
 console.info(context.response.body);

 // Rest of code

 callback();
};
```

2. Save the file.
3. In Admin, go to **System** > **Customization** > **API Extensions** to open the Action Management JSON Editor.
4. Set the `defaultLogLevel` to `info`, which allows low-priority logs to display in Dev Center.

5. Run `grunt` in your project directory to upload your assets with the new logging code.
6. In your sandbox, interact with your storefront so that the relevant action runs its custom function. For the preceding examples, you would add an item to the cart or proceed through the checkout process until the shipping rate information displays.
7. In Dev Center, go to **Logs** > **Action Logs**.
8. Select a tenant and click **OK** to view the logs for that tenant.
9. Identify the logs generated by your function and click them to view details. A lot of the data relevant to applications is formatted as JSON. Kibo recommends you copy this data into an online JSON formatter for easier viewing.

Log Data on a Production Application

While logging data is useful during development, when your application is ready for production, you should make sure to:

1. Remove or comment out console statements from your code.
2. In Admin, go to **System** > **Customization** > **API Extensions** to open the Action Management JSON Editor. Set the `defaultLogLevel` to `error`, which allows only high-priority logs to display in Dev Center.

> Forgetting to set the log level to error will result in performance penalties because low-priority logs generate data more frequently, unnecessarily taxing the system.