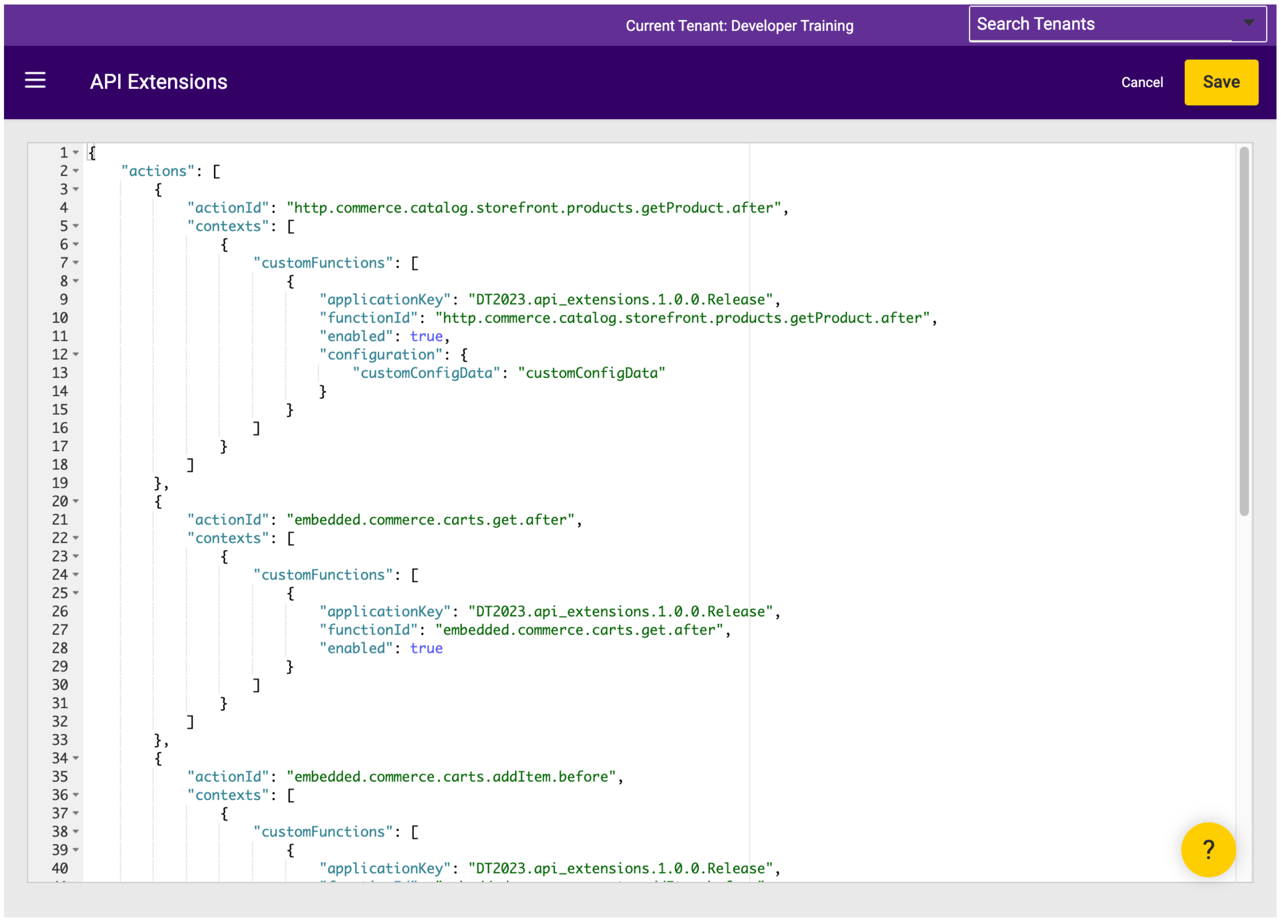
Action Management JSON Editor

The below image demonstrates how the Action Management JSON Editor empowers users to exert precise control over actions through JSON code.

Within Admin, the Action Management JSON Editor allows you to control actions using JSON code. With the Action Management JSON Editor you can:

- Enable or disable the functions bound to actions installed on the sandbox.
- Specify settings, such as timeout and exception behavior, for individual actions.
- Provide custom information to individual actions and to the application that contains the actions.
- Specify logging behaviors associated with actions.



```
{
    "actions": [
        {
            "actionId": "http.commerce.catalog.storefront.products.getProduct.after",
            "contexts": [
                {
                    "customFunctions": [
                        {
                            "applicationKey": "DT2023.api_extensions.1.0.0.Release",
                            "functionId": "http.commerce.catalog.storefront.products.getProduct.after",
                            "enabled": true,
                            "configuration": {
                                "customConfigData": "customConfigData"
                            }
                        }
                    ]
                }
            ]
        },
        {
            "actionId": "embedded.commerce.carts.get.after",
            "contexts": [
                {
                    "customFunctions": [
                        {
                            "applicationKey": "DT2023.api_extensions.1.0.0.Release",
                            "functionId": "embedded.commerce.carts.get.after",
                            "enabled": true
                        }
                    ]
                }
            ]
        },
        {
            "actionId": "embedded.commerce.carts.addItem.before",
            "contexts": [
                {
                    "customFunctions": [
                        {
                            "applicationKey": "DT2023.api_extensions.1.0.0.Release",
```

To open the Action Management JSON Editor:

1. Log in to Dev Center.
2. View a sandbox.
3. In Admin, go to **System** > **Customization** > **API Extensions.**

JSON Structure

With the Action Management JSON Editor, you specify which actions you have installed to a sandbox, the context each action applies to, and the settings an action uses in each context (i.e., the application key, the function you want to execute, and the custom configuration data you want to provide the action). You also specify the custom configuration data available to all actions in the application and the log level that actions use in the application.

The following code block and table demonstrate the options you can configure with the Action Management JSON Editor.

```
{
   "actions": [
      {
         "actionId": "embedded.commerce.carts.deleteCart.before",
         "contexts": [
            {
               "context": 21074,
               "customFunctions": [
                  {
                     "applicationKey": "yourApplicationKey",
                     "functionId": "functionName",
                     "enabled": true,
                     "timeoutMilliseconds": 5000,
                     "exceptionBehavior": "fault",
                     "logLevel": "Info",
                     "configuration": {
                        "yourCustomField": "value"
                     }
                  }
                  ...
               ]
            }
            ...
         ]
      }
      ...
   ],
   "configurations": [
      {
         "applicationKey": "yourApplicationKey",
         "configuration": {
            "yourCustomField": "value"
         }
      }
      ...
   ],
   "defaultLogLevel": "Info"
}
```

`defaultLogLevel`	Specifies which types of application logs display in Dev Center, based on priority level. Possible values mirror Apache's log4net: `All` , `Debug` , `Info` , `Warn` , `Error` , `Fatal` , and `Off` . When deploying an API Extensions application to production, set this value to `Error` to avoid performance penalties.
actions	An array of actions.
`actionId`	Identifies a specific action. This ID matches the naming conventions in the `assets/functions.json` file.
actions[contexts]	The per-site settings that apply to an individual action.
`context`	(Optional) The siteId for the site you want to apply the nested settings to. You can omit this field if you want to apply the same settings to an action across all your sites.
actions[contexts[customFunctions]]	An array of custom functions tied to an action. Some actions can run only one function, but other functions can run multiple functions.
`applicationKey`	The application key of the API Extensions application.
`functionId`	The name of the custom function tied to the action, per the naming conventions set in the manifest files located in the `assets/src` directory.
`enabled`	(Optional) A Boolean that controls whether the function is enabled or disabled. The default is `true` .
`timeoutMilliseconds`	(Optional) The number of milliseconds that the function waits before timing out. The default is `5000` milliseconds.
`exceptionBehavior`	(Optional) The behavior to take when an error is encountered, either `fault` or `continue` . The default is `fault` .
`logLevel`	(Optional) Specifies which types of function-specific logs display in Dev Center, based on priority level. Possible values mirror Apache's log4net: `All` , `Debug` , `Info` , `Warn` , `Error` , `Fatal` , and `Off` . When deploying an API Extensions application to production, set this value to `Error` to avoid performance penalties.
actions[contexts[customFunctions[configuration]]]	Custom function-level settings that you can create. If you create custom settings with the same names as custom settings created at the application level, these settings take precedence over the application-level configurations.
`yourCustomField`	Custom object data.
configurations	Custom settings that apply to all actions in the API Extensions application.

`applicationKey`	The application key of the API Extensions application.
configurations[configuration]	Custom application-level settings that you can create. If you create custom settings with the same names as custom settings created at the function level, these settings are overwritten by the function-level configurations.
`yourCustomField`	Custom object data.