What You Can Do With API Extensions

API Extensions are a development framework built by leveraging the Node JS runtime that allows server-side JavaScript applications to execute on the platform. With API Extensions, you can make changes to the platform by manipulating API operations through the HTTP request-response protocol as well as interacting directly with underlying Kibo microservices.

Example: Keeping Track of a Shopper's Recurring Purchases

The following JavaScript code shows a high-level example of how you can use API Extensions to leverage the lodash package and the Node.js SDK to keep a record of products that are repurchased frequently by a shopper and store those products in a custom attribute within the shopper's account. The example uses the `embedded.commerce.order.action.after` action available in the `commerce.orders` domain of the API Extensions framework.

```
// Import awesome libraries
var _ = require('lodash');
var OrderClient = require('mozu-node-sdk/clients/commerce/orders');
var CustomerHelper = require('./customer-helper');

// Run this function after an order is placed (corresponds to the 'embedded.commerce.order.action
.after' action)
module.exports = function(context, callback) {

    // Access the context argument to retrieve useful Mozu data
    var orderClient = new OrderClient(context);
    var customerHelper = new CustomerHelper(context);
    var order = context.get.order();

    orderClient.getOrders({ pageSize: 50 }).then(function(last50UserOrderHistory) {
        // Get the list of every product code a shopper has ever ordered and create a custom attribut
e on the shopper's account that lists all the product codes that have been re-purchased on separat
e orders
        var productCodesOrderedOnceBefore = _.intersection(_.pluck(order().items, 'product.product
Code'), _.pluck(_.flatten(_.pluck(last50UserOrderHistory, 'items'), 'product.productCode')));

    if (productCodesOrderedOnceBefore.length > 0) {
        return customerHelper.updateAttribute({
            accountId: context.get.order().customerAccountId,
            attribute: 'tenant~repurchasedProducts'
            values: productCodesOrderedOnceBefore
    });
    }
    }).then(function() {
        // Success!
        callback();
    }).catch(callback);
}
```

Take Full Control of the Shopper Experience

API Extensions gives you tremendous control over the commerce experience in a fully managed multi-tenant SaaS environment. Developers can use API Extensions applications to augment, replace, or customize the behavior of Kibo storefronts and API microservices, resulting in richer, more rewarding experiences for your shoppers. For example, you can:

- Enable unique rewards in response to a shopper adding an item to a cart.
- Provide shoppers with incentives to share your site on social media during the checkout process.
- Create soft allocations on a cart.
- Create custom shopper UIs and web applications.

Empower Developers to Innovate

API Extension applications work by giving you the ability to bind custom JavaScript functions to actions so you can surface unique behaviors throughout the platform. With API Extensions, you can run an application transactionally and augment data during the transaction process instead of

being limited to listening for events or receiving standard information over HTTP. The following are examples of the limitless innovations you can achieve with API Extensions to benefit your business users:

- Integrate third-party shipping and logistics providers with Kibo.
- Customize how Kibo processes orders and returns.
- Create custom inventory handling logic.
- Integrate a third-party search engine to replace the Kibo SOLR implementation.

Write Your Code in a Simple, Universal Language

With API Extensions, there's no need to learn a new language. API Extension applications are Node.js compatible and allow you to leverage the thousands of NPM packages developed by the community to enhance the functionality of applications built on Kibo.

See the Getting Started with API Extensions guide for a quickstart tutorial.

API Extension Ideas

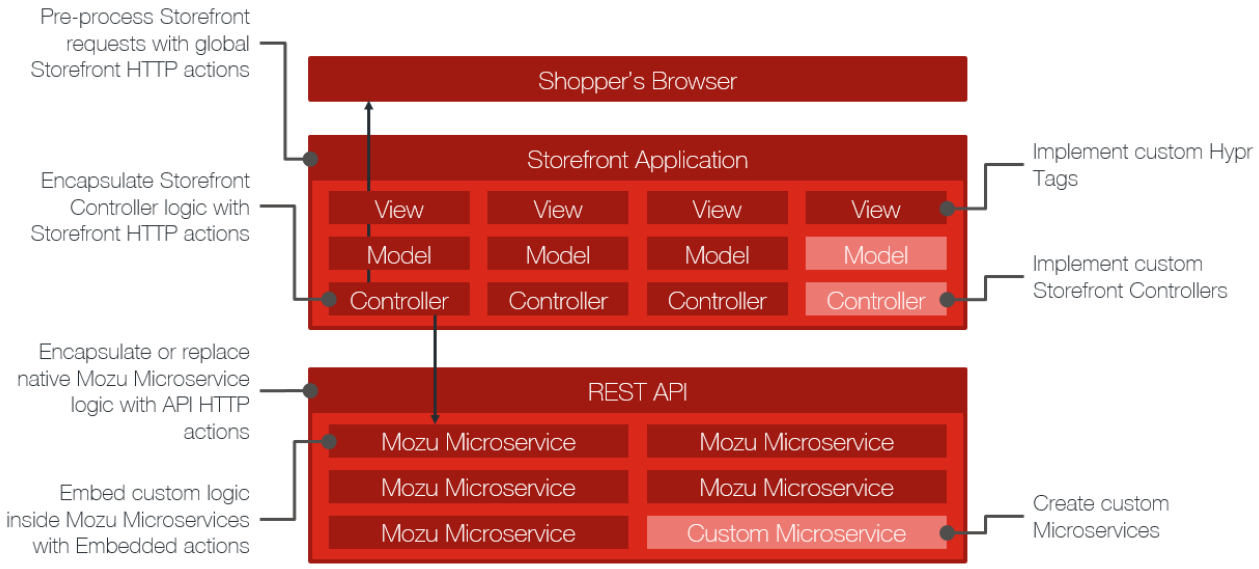
API Extensions are a development framework that allows you to run custom JavaScript functions when actions occur. For example, you can run custom functions whenever a shopper creates a new account, deletes an item from a cart, or views a page on your site. Custom functions augment or bypass out-of-the-box behavior and provide your shoppers with a more tailored and compelling shopping experience. The following are just a few examples of what you can do with API Extensions:

- Retrieve custom shipping rates based on the shopper's location and choice of shipping method when shipping rates are requested during the checkout process.
- Create soft allocations on a cart. Whenever a shopper adds an item to a cart, an action reserves the item for a specified amount of time.
- Provide shoppers with loyalty rewards based on purchase criteria of your choice.
- Create custom coupon validation logic.
- Create custom product validation logic.
- Integrate a third-party search engine to replace the Kibo SOLR implementation.
- Create custom search processing logic.
- Create custom inventory handling logic.
- Integrate custom shipping and logistics providers with Kibo.
- Create custom web applications.
- Integrate with a third-party CMS.
- Customize how Kibo processes returns.
- Customize how Kibo processes orders.

- Create "Bonus Item" logic for automatically adding items to the cart under certain conditions.
- Dynamically segment customers based on site usage patterns.
- Process imported passwords from legacy systems.
- Create a custom customer qualification service and registration UI.
- Integrate with a third-party blog platform.

How API Extensions Fit into the Platform

As shopper, merchant, and application actions occur, Kibo invokes the API customize the actions to your needs. API Extensions are a growing framework that continually adds action entry points to give you an ever more complete ability to customize the platform. The following diagram provides a high-level overview of the typical components that the API Extension framework will interact with.



Major components of API Extensions are listed below.

- **API Extensions Runtime:** The runtime is embedded in the platform and uses the Google V8 JavaScript engine to execute API Extension applications.
- **API Extension applications:** These applications contain the custom JavaScript functions that execute whenever an action occurs. API Extension applications are Node.js compatible and can be developed and tested on local development environments before being uploaded to Dev Center. API Extension applications leverage:
  - the Node.js SDK for accessing the REST API.
  - the distributed cache layer to create scalable interactions.
  - the logging infrastructure for debugging, informational, and error reporting purposes.

    - packages from the Node.js ecosystem, providing developers access to thousands of existing code libraries.
- **API Extension development tools**: These tools include a Yeoman scaffolder that simplifies the process of generating application assets. The tools also leverage Grunt to build and synchronize application assets with the cloud.

API Extensions can also customize the following components.

- **API:** Modify or entirely short-circuit inbound API requests and outbound responses.
- **Microservices:** Enrich or entirely replace Kibo eCommerce core capabilities in order to integrate best-of-breed technologies or your own custom solutions.
- **Storefront Routing Engine:** Bind custom functions to every page request on the storefront.

How do I use API Extensions?

First you use JavaScript to code the custom functionality that you want to associate with an action(s) in Kibo. Then, you upload the action(s) to an application in Dev Center and install that application to a sandbox. Finally, you enable the action(s) from within the sandbox using the Action Management JSON editor. To get started, we suggest the following learning path:

1. Learn about the basic structure of an API Extension application.
2. Learn about the types of actions available for you to bind custom functions to.
3. Learn about the JavaScript templates you use to code the custom functionality for an action.
4. Learn about the JSON editor you use to enable actions for a sandbox.
5. Work through the Quick Start for a start-to-finish example of implementing an API Extension application.
6. Leverage the reference content to learn the details of actions and the methods they have access to.

For information on how to implement custom Hypr tags with API Extensions, see the "Send Custom Data to Your Theme" section of the Programming Patterns guide.