Application Development Best Practices

This topic describes best practices and answers common questions for third parties developing applications for the Kibo Composable Commerce Platform (KCCP). This topic assumes you are familiar with basic Application Development Requirements.

Security

The following sections describe best practices for protecting your applications and users from security threats. For best practices specific to application behaviors, see the behaviors guide.

Encrypt Sensitive Data

Always encrypt any sensitive data that your application might handle in transit or at rest. Failure to encrypt the following data may prevent Kibo from approving your application when you submit it for certification:

- Application keys/IDs
- Shared secrets
- Usernames
- Passwords
- Access keys

Use a Configuration Dialog to Get User Credentials

As an application developer, you should never directly request authentication credentials from a merchant for either Kibo or a third-party platform. For this reason, KCCP provides a built-in configuration dialog that is essentially an iframe into which you can load any HTML. You can use this dialog to allow merchants to enter their own credentials. Then, you are only responsible for encrypting the data and ensuring it isn't logged or stored in a sensitive location.

To create a configuration dialog:

1. Write the HTML required for any configuration data and add it to your application code.
2. Upload your application to Dev Center.
3. Click **Develop** > **Applications**.
4. Double-click the application you want to configure.
5. Click the **Packages** tab.
6. Enter the URL for the content you want the configuration dialog to display in the **Configuration URL** field. This causes the Configuration link to appear when users access the application from any tenant to which it is installed.

OAuth: Redirect Users Back to Your Application

Many platforms you might want to integrate with use the OAuth authorization standard. You cannot implement OAuth in an iframe across domains, so you must temporarily take the user out of your configuration dialog while they authenticate with the third-party.

In cases where OAuth is required, you can use redirects to return the user to Admin and the configuration dialog. Simply direct the user back to the tenant URL with `/#configure` appended. For example:

```
https://t00000.mozu.com/Admin/s-1111/capability/edit/fbe22a718c7e3245a16543210c1bd334/#configure
```

Prevent Cross-Tenant Access

By default, all URLs contain a tenant ID. To prevent potential misuse of your tenant ID, you should configure your application to transform it. For example, when developing applications, the Kibo Integrations team hashes tenant IDs with session IDs to conceal the tenant ID and prevent unauthorized access.

Event Management

Keep the following information in mind when dealing with events:

- When you configure an application to subscribe to events, Kibo sends an HTTP POST request to your application server whenever that event occurs (e.g., product updated). Refer to Event Subscription for more information.
- The platform sends one event for each site and catalog you have. For example, if you have two sites, one master catalog, and two catalogs, it sends five POST requests for each event to which your application is subscribed.
- Whenever you make changes to application behaviors or event subscriptions, you must re-install the application on your sandbox and re-enable it in Dev Center in order for the changes to work.

Verify Event Authenticity

Although optional, it's a good idea to verify that an event originated from the platform before interacting with it programmatically. To verify the authenticity of an event, your application must generate a SHA 256 hash that matches the SHA 256 hash generated by the platform.

Here’s a description of how the hashing function works:

1. Use Base64 encoding.
2. Concatenate the application's shared secret with itself to get a starting key.

3. Generate a SHA 256 hash of the starting key.
4. Concatenate the Base64-encoded hash, event date from the request header, and request body.
5. Generate a second SHA 256 hash.
6. Encode the second SHA 256 hash using the Base64 encoding scheme.
7. Compare the SHA 256 hash in the request header (`x-vol-hmac-sha256";`) with the hash your application generated. Matching hashes confirms event authenticity.

Tips:

- Use the built-in hashing function in the Node, .NET, and Java toolkits. You'll have to write your own if you're using another language.
- Enhance the security of your application by enforcing a time constraint on generating the SHA 256 hashes. Make sure you're application server is synchronized with the platform by using the NIST Internet Time Service.

Here's an example of a POST request (header and body) coming from KCCP:

```
x-vol-correlation: 4e84d4304b6342a4a32eb0e9efba9a87
x-vol-tenant: 12345
x-vol-currency:
x-vol-locale:
x-vol-tenant-domain: t12345.sandbox.mozu.com
x-vol-site: 16772
x-vol-catalog: 3
x-vol-master-catalog: 1
Date: Tue, 01 Mar 2016 20:51:22 GMT
x-vol-hmac-sha256: n8R65NCMOoemLohdg0uMMZVdzOcJFrcUngon08D3e/g=
Content-Type: application/json; charset=utf-8
Host: 702443ca.ngrok.io
Content-Length: 211
X-Forwarded-Proto: https
X-Forwarded-For: 162.219.105.124
```

```
{
   "eventId": "a585728e-18eb-49af-afea-a5bc0157b267",
   "topic": "customeraccount.updated",
   "entityId": "1001",
   "timestamp": "2016-03-01T20:51:37.4784068Z",
   "correlationId": "4e84d4304b6342a4a32eb0e9efba9a87",
   "isTest": "false"
}
```

Acknowledge Events

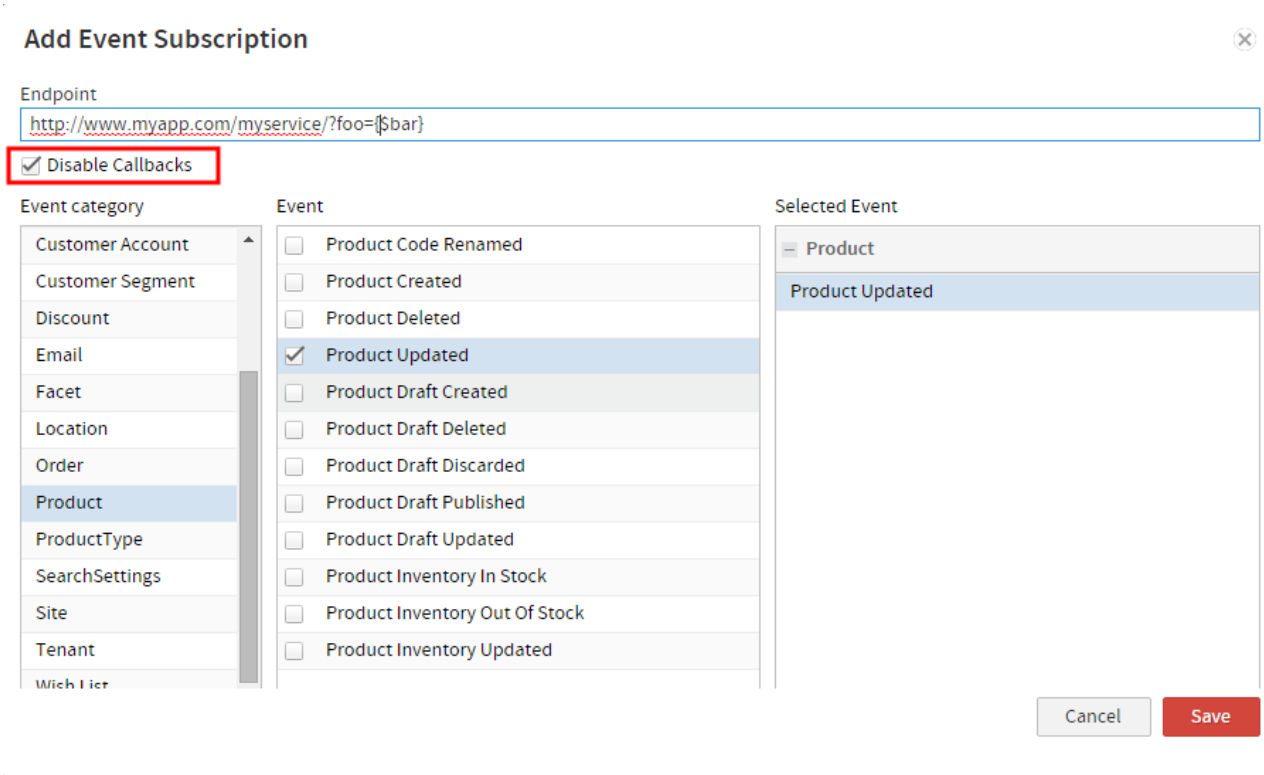
KCCP is expecting an HTTP status code in the 200 range in response from your application, but it's important to acknowledge unsuccessful POST requests as well. You should send status codes back as soon as possible, otherwise, KCCP will try to resend the event after 30 seconds and then continue resending the event according to the schedule defined here for up to 24 hours.

KCCP sends a unique ID for each event it tries to resend in the response body, even if it's the same event.

Prevent Infinite Callback Loops

Infinite loops can occur when your application subscribes to an event that can be triggered by events in a third-party system. For example, suppose your application subscribes to `product.updated` events. So, when you update a product your application updates a product in a third-party system, which triggers another `product.updated` event.

To prevent infinite loops related to event subscriptions, make sure that the **Disable Callbacks** checkbox is checked when you add the event subscription to your application in Dev Center.



Click here for instructions on adding an event subscription to your application.

Performance Optimization

The following sections contain best practices for improving the performance of your applications.

Store Data in the Database (MZDB)

To use the MZDB effectively, you must have a general understanding of entity lists and entities. Entity lists are similar to database tables—describing the types of data that can be stored, which properties should be indexed for high-scale retrieval, and the list's read/write security model. Currently, you can create entity lists with the API or API Extension applications. Entities are objects in an entity list and are similar to rows in a database table, however, objects are rich JSON

structures rather than fixed tabular rows. You can use entity lists to store and retrieve website content throughout the platform and third-party applications. For example, creating and maintaining a list of physical store locations.

You can create as many fields as you want in a custom entity list, but you can only index a maximum of four. Indexing fields impacts application performance in different ways. There's a correlation between application performance and number of indexed fields in an entity list:

- Indexed fields *increase* the speed of read operations.
- Indexed fields *decrease* the speed of write operations.

So, an application that performs mostly read operations benefits from a higher number of indexed fields, but an application that performs mostly write operations will have slower performance as the number of indexed fields increases.

Limit Data Returned from API Calls

If you're developing an application that calls large JSON objects from the API, or if you're doing targeted reporting (e.g. product pricing), you can use the `responseFields` URL parameter to filter the data returned inside a JSON object. Efficiently allocating application memory will help improve performance.

Only use the responseField parameter to retrieve data. Attempting to update data using this parameter may cause data loss.

For example, `commerce/catalog/storefront/products/{product code}` returns a JSON object that (for a specified product code) looks like this:


```
{
   "productCode":"1005",
   "productSequence":"5",
   "productUsage":"Standard",
   "fulfillmentTypesSupported":[
      "DirectShip",
      "InStorePickup"
   ],
   "goodsType":"Physical",
   "content":{
      "productName":"Piona Kailas Patent Pump",
      "productFullDescription":"\"Faux patent leather upperAlso available in a faux metallic & print
ed leather upper Ankle strap with an adjustable buckle\1 hidden platform\5 heel\Synthetic sole",
      "productShortDescription":"You'll be hot to trot in the Kailas by Piona. This sexy pump is is su
re to draw some attention.",
      "metaTagTitle":"",
      "metaTagDescription":"",
      "metaTagKeywords":"",
      "seoFriendlyUrl":"",
      "productImages":[
         {
         "imageUrl":"/files/64/1/1e3813e5-e60a-462b-a109-087682eb2a31",
         "sequence":"1"
```

```
        }
      ]
      },
  "purchasableState":{
      "isPurchasable":"true"
      },
  "isActive":"true",
  "publishState":"Live",
  "price":{
      "price":"60",
      "priceType":"List",
      "catalogListPrice":"60"
  },
  "productType":"Shoe_Women",
  "productTypeId":"3",
  "isTaxable":"true",
  "pricingBehavior":{
      "discountsRestricted":"false"
      },
  "inventoryInfo":{
      "manageStock":"false"
      },
  "createDate":"2013-12-17T05:06:15.980Z",
  "dateFirstAvailableInCatalog":"2013-12-17T05:06:15.980Z",
  "daysAvailableInCatalog":"811",
  "categories":[],
  "measurements":{
      "packageWeight":{
      "unit":"lbs",
      "value":"1.25"
        }
      },
  "properties":[
        {
      "attributeFQN":"tenant~availability",
      "isHidden":"false",
      "isMultiValue":"false",
      "attributeDetail":{
      "valueType":"Predefined",
      "inputType":"List",
      "dataType":"String",
      "usageType":"Property",
      "dataTypeSequence":"1",
      "name":"Availability",
      "searchableInStorefront":"true",
      "allowFilteringAndSortingInStorefront":"true"
      },
  "values":[
        {
      "value":"24hrs",
      "stringValue":"Usually Ships in 24 Hours"
        }
      ]
      }
  ]
}
```


If you only need the call to return a product's name and description, you can specify that in the response field. For example, `commerce/catalog/storefront/products/{product code}?`

`responseFields=content(productName, productShortDescription)` returns a JSON object that looks like this:

```
{
   "content": {
      "productName": "Piona Kailas Patent Pump",
      "productShortDescription": "You'll be hot to trot in the Kailas by Piona. This sexy pump is is su
re to draw some attention."
      },
   "isTaxable": "true",
   "createDate": "2013-12-17T05:06:15.980Z"
}
```

Tips:

- Access all nested fields inside an object property by specifying the property only. For example, `?responseFields=property`.
- Use parentheses to access specific nested fields inside an object property. Use a comma-sparated list to select multiple fields. For example, .
- Access multiple nested fields inside multiple object properties using the following syntax (comma-separate properties): `?=responseFields=property1(field1, field2), property2(field3, field4)`.
- When working with collection-based API endpoints (e.g., GetProducts vs. Get Product), you must use a different syntax to access fields inside JSON object properties. The GetProducts endpoint contains an `items` property that you must specify as the first property. For example, `?responseFields=items(content(productName), price(priceType), productCode)`.