Java SDK

When developing applications, you can install the Java SDK to authenticate with Kibo and send API requests using Java. This SDK has access to the full functionality of Kibo's REST APIs.

For more detailed information about application development, refer to the Application guides. This guide describes how to expand upon an application by adding an SDK.

Before You Begin

You must have the following Kibo access details:

- Maven: a tool for building and managing Java-based projects.
- A Dev Center Account with a provisioned sandbox.
- An Application Key (also called the Client ID) and Secret.
- A Kibo tenant.

Get Started

This tutorial demonstrates installing the Java SDK, authenticating with Kibo credentials, and making an API call. The first example is of a console application that retrieves product details, while the follow-up example retrieves the number of customer accounts for a Kibo eCommerce site. Use these concepts to build a fully-fledged application of your own.

Create an Application

First, create an application in the Dev Center with the appropriate behaviors:

1. Log in to Dev Center.
2. Create a new application.
3. Add the Customer Read Behavior to the application. This step is necessary to give your application the necessary permissions to read customer accounts. If you design additional functionality for your application, such as updating an order, add the appropriate behaviors to avoid a permissions error.
4. Install the application to the sandbox of your choice.
5. Enable the application in Admin. If you decide to add additional behaviors to your application after this step, you must reinstall the application to your sandbox and re-enable the application in Admin to apply the new behaviors.
6. Locate your Application Key, Shared Secret, Tenant ID, and Site ID. Obtain the Application Key and Shared Secret from the application details page in the Dev Center. Obtain the Tenant ID and Site ID by viewing your site and looking at the URL, which has the

format `t{TenantID}-s{SiteID}.sandbox.mozu.com`.

- You can also use the Get Tenant API call to return the Site ID and a Master Catalog ID if needed.

Install the Java SDK

Use Maven to install the Java SDK and identify any API modules you intend to use as dependencies. The SDK contains a module for each API domain, similar to the organization in the Kibo API documentation. You can find these modules in the kibocommerce folder of the SDK.

This example uses the Catalog Administration API module.

```
<properties>
   <kibocommerce.version>2.0.0-SNAPSHOT</kibocommerce.version>
</properties>

<dependencies>
   <dependency>
      <groupId>com.kibocommerce</groupId>
      <artifactId>catalogadministration</artifactId>
      <version>${kibocommerce.version}</version>
   </dependency>
   <!-- ...other Kibo Service Modules -->
</dependencies>
```

Configuration

Create a Kibo Configuration object with your authentication values that can be re-used for all API clients.

1. Create a `.env` file at the root of your project and enter your credentials and any other required values such as Catalog IDs.

   ```
   KIBO_TENANT=
   KIBO_SITE=
   KIBO_CLIENT_ID=
   KIBO_CLIENT_SECRET=
   KIBO_CATALOG=
   KIBO_MASTER_CATALOG=
   KIBO_LOCALE=
   KIBO_CURRENCY=
   KIBO_HOME_HOST=
   KIBO_TENANT_HOST=
   KIBO_PCI_HOST=
   KIBO_DEBUG_CLIENT=
   ```

2. Build the configuration:

```
import com.kibocommerce.sdk.common.ApiCredentials;
import com.kibocommerce.sdk.common.KiboConfiguration;


//...
public KiboConfiguration getConfiguration() {
   return KiboConfiguration.builder()
            .withTenantId({tenantId})
            .withSiteId({tenantId})
            .withCredentials(
                  ApiCredentials.builder().setClientId("client_id")
                        .setClientSecret("client_secret").build())
            .withTenantHost("t{tenantId}.sandbox.mozu.com")
            .withHomeHost("t{tenantId}.sandbox.mozu.com")
            .build();
}
```

3. To initialize the configuration from your system properties:

```
import io.github.cdimascio.dotenv.Dotenv;
import com.kibocommerce.sdk.common.KiboConfiguration;


public KiboConfiguration getConfiguration() {
   // using dotenv to load .env file into system properties
   Dotenv dotenv = Dotenv
      .configure()
      .systemProperties()
      .load();
   // initialize configuration object from system properties
   KiboConfiguration configuration = KiboConfiguration.builder()
            .fromSystemProperties()
            .build();
   return configuration
}
```

Create an API Client

Create an API client that makes calls with the appropriate module(s). This client makes a Get Product API call to retrieve product details:

```
// Import Kibo Configuration
import com.kibocommerce.sdk.common.ApiCredentials;
import com.kibocommerce.sdk.common.KiboConfiguration;
// Import Kibo Commerce Catalog Administration API
import com.kibocommerce.sdk.catalogadministration.api.ProductsApi;
// Import the Kibo Commerce Catalog Administration Models
import com.kibocommerce.sdk.catalogadministration.models.CatalogAdminsProduct;


public final class App {
   private App() {
   }

   public static void main(String[] args) {
      // Get Kibo Configuration Reference
      KiboConfiguration configuration = getConfiguration();
      // Build API Instance
      ProductsApi api = ProductsApi.builder().withConfig(configuration).build();
      // Make API call to Catalog Administration service to get product
      CatalogAdminsProduct product = api.getProduct(productCode);
   }
}
```

Customer API Example

This additional example demonstrates a Java project that calls the Customer API module to retrieve a list of customers and display the total customer amount.

1. Using the terminal, generate an example project using Maven.

   ```
   mvn archetype:generate -DarchetypeArtifactId="maven-archetype-quickstart" -DarchetypeGr
   oupId="org.apache.maven.archetypes" -DarchetypeVersion="1.4" -DgroupId="com.example
   " -DartifactId="kibodemo"
   ```

2. Open the new pom.xml file and add Kibo's customer module as a dependency.

   ```
   <dependencies>
      <dependency>
        <groupId>com.kibocommerce</groupId>
        <artifactId>customer</artifactId>
        <version>2.0.0</version>
      </dependency>
      <!-- ...other Kibo Service Modules -->
   <dependencies>
   ```

3. Open App.java and import the Kibo dependencies.

   ```
   // App.java
   import com.kibocommerce.sdk.common.KiboConfiguration;
   import com.kibocommerce.sdk.common.ApiException;
   import com.kibocommerce.sdk.customer.api.CustomerAccountApi;
   import com.kibocommerce.sdk.customer.models.CustomerAccountCollection;
   ```

4. Inside the main method, configure the API client and make the call to retrieve customer accounts.

```
try {
  // Configure the Kibo SDK with your Kibo Commerce tenant credentials
  KiboConfiguration configuration = KiboConfiguration.builder().build();

  // Initialize the CustomerAccountApi with your Configuration.
  CustomerAccountApi customerAccountApi = new CustomerAccountApi(configuration);

  // Fetch 10 customer accounts to retrieve a list of customers
  CustomerAccountCollection customers = customerAccountApi.getAccounts(0, 10, null, null,
null, null, null, null, null);

  // Display the total number of customers
  System.out.println("Total Customers: " + customers.getTotalCount());
} catch (ApiException e) {
  // Handle the exception
  System.out.println("Error in getting customers: " + e.getMessage());
}
```

5. Run the application.

The final App.java file should look like this:

```
import com.kibocommerce.sdk.common.ApiCredentials;
import com.kibocommerce.sdk.common.ApiException;
import com.kibocommerce.sdk.common.KiboConfiguration;
import com.kibocommerce.sdk.customer.api.CustomerAccountApi;
import com.kibocommerce.sdk.customer.models.CustomerAccountCollection;


/**
 * Hello world!
 */
public class App {
    public static void main( String[] args ) {
        try {
            // Use your Application Key and Application Secret
            ApiCredentials credentials = ApiCredentials.builder()
                            .setClientId("client_id")
                            .setClientSecret("client_secret")
                            .build();

            // Configure your Tenant ID, Site ID, and Hostname
            KiboConfiguration.builder()
                .withTenantId(12345)
                .withSiteId(12345)
                .withCredentials(credentials)
                .withTenantHost("t39368.sandbox.mozu.com")
                .withHomeHost("t39368.sandbox.mozu.com")
                .build();

            // Configure the Kibo SDK with your Kibo Commerce tenant credentials
            KiboConfiguration configuration = KiboConfiguration.builder().build();

            // Initialize the CustomerAccountApi with your Configuration.
            CustomerAccountApi customerAccountApi = new CustomerAccountApi(configuration);

            // Fetch 10 customer accounts to retrieve a list of customers
            CustomerAccountCollection customers = customerAccountApi.getAccounts(0, 10, null, null,
 null, null, null, null, null);

            // Display the total number of customers
            System.out.println("Total Customers: " + customers.getTotalCount());
          } catch (ApiException e) {
            // Handle the exception
            System.out.println("Error in getting customers: " + e.getMessage());
          }
    }
}
```