Application Development Requirements

The Kibo Composable Commerce Platform is designed for extensibility. Through our REST API, you have complete access to the operations and entities you need to develop applications. We also provide SDKs that you can use to write apps in the language of your choice.

Advantages of SDKs

Kibo strongly recommends using one of our SDKs for app development. If you use an SDK, all of the following tasks are handled behind the scenes by the SDK:

- Routing requests to proper endpoint URLs
- Ensuring all calls to use the correct API context
- Authenticating your app
- Creating and regenerating access and refresh tokens
- JSON conversion
- Event handling and decrypting/validating event message

The remainder of this topic explains these concepts as they pertain to all applications. Refer to the readmes and code comments for each SDK for more specific information.

Requests to the API

To facilitate application development, the API for all platform/application-level services is hosted behind a generic US or EU domain. You can make calls to either a production or sandbox environment, as well as optionally include a Site ID in addition to the required Tenant ID (which would make the Base URL `t10000-s00000` instead of `t10000`). The tp0 is your tenant's assigned production pod.

- Example US Sandbox Tenant: `https://t10000.sandbox.mozu.com/api`
- Example US Production Tenant: `https://t10000.tp0.mozu.com/api`
- Example EU Sandbox Tenant: `https://t100000.sb.euw0.kibocommerce.com/api`
- Example EU Production Tenant: `https://t100000.tp0.euw1.kibocommerce.com/api`

Your application can send requests to this domain by including it in the request path. For example, you can use the following path to send a production authentication request:

```
http://t00000.tp0.mozu.com/api/platform/applications/authtickets
```

Requests require the following:

- The authentication ticket your application uses to complete API calls
- The resource URI you want to query

- The API context in the request header
- The application must have the appropriate behaviors to access the API

API Context

For any request your application sends, the request header must include an API context which often identifies the site and catalog you are referencing. If you use the hostname format that includes the site ID (t00000-s00000.tp0.mozu.com) then the master catalog, catalog, site, locale, and currency context are inferred by the site and do not need to be explicitly provided. Likewise, the tenant is usually not necessary in the header since the tenant ID is already included in the hostname.

The following is an example explicitly defining that context:

```
x-vol-tenant: 0000
x-vol-master-catalog: 1
x-vol-catalog: 1
x-vol-site: 11111
```

Supported Headers

If you are using one of the SDKs to develop your application, you can look at the ApiContext and Headers files at the top level of the source directory to see how the API context is implemented and what headers you can include in your requests. Otherwise, refer to the API documentation for a full list of possible headers.

Authentication

Any application that calls into the API must authenticate. When you create an application in the Dev Center, an Application Key/ID and Secret is generated. You use these values to authenticate, which will return your access and refresh tokens.

See the Application Authentication API specs for more details about the auth operations: there is a standard auth call and an OAuth 2.0 JWT call, which have slightly different requests and responses.

SDKs also include logic that handles authentication for you using your API context and the Application Key and Shared Secret. For example, if you are using the .NET SDK, you can add this information to the app.config file:

```
<configuration>
  <appSettings>
    <add key="ApplicationId" value="AppKeyFromDevCenter" />
    <add key="SharedSecret" value="SharedSecretFromDevCenter" />
  </appSettings>
  ...
</configuration>
```

Access and Refresh Tokens

An **access token** establishes an application's identity. When an app calls a specific API operation, the access token is passed as part of the application claims information in the request header.

Refresh tokens allow an application to refresh an expired access token without re-authenticating the entire application. After the access token expires, the refresh token can use the Application Key and Shared Secret to generate a new authentication ticket that contains the same refresh token and refresh token expiration, but contains a new access token and access token expiration.

Both access tokens and refresh tokens are subject to expiration. After the refresh token expires, you must generate a new authentication ticket. Access tokens are viewable and readable for external programs. Refresh tokens should never be shared. Always protect information about your refresh token and shared secret.

Generate an Authentication Ticket

1. Copy your Application Key and Shared Secret from Dev Center. Go to **Develop** > **Applications** and double-click the app you are developing to view this information.
2. In your external application, run a POST operation to the platform/applications/authtickets or platform/applications/authtickets/oauth resource. You can do this to the sandbox.mozu.com host for testing on a sandbox environment, while your complete and certified app will make requests using the production URL.
3. In the request body, enter the Application Key/ID and Secret. All current applications use the full Application Key value for the ID. Legacy applications use the Application ID instead of the full Application Key. The Application ID can be found as part of the full Key: `<dev account namespace>.<application ID>.<application version>.<package name>` .

The system returns the refresh token, access token, and expiration information.

Refresh an Authentication Ticket

If you used the platform/application/authtickets method and the access token has expired but the refresh token is still valid, complete the following steps to refresh the authentication ticket:

1. Run a PUT operation to the platform/applications/authtickets/refresh-ticket URL.
2. In the request, enter the refreshToken value string.

The system returns the refresh token, access token, and new expiration information.

Run Operations from an Application

After you provision a sandbox, install an application in that sandbox, and generate an authentication ticket for the application, you can begin to run API operations using the application.

Running an operation in the API consists of the following procedures:

- Retrieve the tenant ID
- Retrieve the domain name for the URI endpoint
- Use the tenant URL to perform API operations

Retrieve the Tenant ID

You can view the tenant ID for a sandbox in the sandbox URL in Admin, or you can get it programmatically using the API:

1. In the Tenants resource, run a GET operation using the following URL: `http://t00000.tp0.mozu.com/api/platform/tenants`
2. In the request body, note the tenantID value.

Retrieve the Domain Name for the URI Endpoint

To retrieve the domain name the application uses to run API operations in a development store, complete the following procedure:

1. In the application, in the Tenants resource, run a GET operation using the following URI: `http://t00000.tp0.mozu.com/api/platform/tenants/{tenantid}`
2. In the request URI, enter the tenant ID you retrieved in the previous procedure.
3. In the request body, note the following information:
    - The DomainName value for the tenant, which appears as: `{tenantID}.{hostname}.mozu.com`
    - The master catalog ID values for the tenant.
    - The catalog ID values for the tenant.
    - The site ID values for the tenant.

Use the Tenant URL to Perform API Operations

To use the sandbox domain name for the development store to begin performing API operations using the application, complete the following procedure:

1. In the application, run the operation you want to perform, using the URL retrieved in the previous procedure: `{domainname}/api/{resourcepath}`
2. In the request header, specify the API context.