Delete Package (Before)

This action manipulates the HTTP request or response before the DeletePackage operation occurs in Kibo.

<table>
<thead>
<tr>
<th>Action Type</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Action ID</td>
<td>http.commerce.returns.deletePackage.before</td>
</tr>
<tr>
<td>Runs multiple custom functions?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

JavaScript File Structure

Action files share the following basic structure:

```javascript
module.exports = function(context, callback) {
  // Your custom code here
  callback();
};
```

When you code the custom function for an action, you have access to two arguments:

- `callback` — This argument follows the established JavaScript callback pattern: it takes an error as the first argument (or null if there is no error) and a result as the second argument (if required).
- `context` — This argument provides the function access to relevant objects and methods that interface with Kibo.

Context: HTTP

The following methods and objects are available to this action through the use of the `context` argument.

**REST API Operation**

This action corresponds to the `commerce/returns/packages/deletePackage` operation.

**HTTP Request**

DELETE `api/commerce/returns/{returnId}/packages/{packageId}`

**Request Body**

No request body content for this action.

**Response Body**

No request body content for this action.

**Context Objects Available to All HTTP Actions**

- `request`
- `response`
Context Methods Available to All HTTP Actions

- request.get
- response.get
- response.set
- response.set
- response.remove
- response.redirect
- response.end
- get.resource
- get.resourceStatus
- items.urlHelper.getUrl

Context Objects Available to All Actions

- apiContext
- configuration

Context Objects Available to All HTTP Actions

**request**

Accesses the current HTTP request. In the case of Before actions, updates can be made to the request before Kibo processes it.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>params</td>
<td>object</td>
<td>The input parameters to the service or webpage.</td>
</tr>
<tr>
<td>headers</td>
<td>object</td>
<td>The request headers.</td>
</tr>
<tr>
<td>method</td>
<td>string</td>
<td>The request method.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>The request URL.</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>The request path.</td>
</tr>
<tr>
<td>cookies</td>
<td>object</td>
<td>The request cookies.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Available only for Storefront HTTP actions.</em></td>
</tr>
<tr>
<td>query</td>
<td>object</td>
<td>The request query.</td>
</tr>
<tr>
<td>Properties</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>href</td>
<td>string</td>
<td>The request href.</td>
</tr>
<tr>
<td>secure</td>
<td>Boolean</td>
<td>Indicates whether the request uses HTTPS.</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>The request IP address.</td>
</tr>
<tr>
<td>ips</td>
<td>string</td>
<td>The request secure IP address.</td>
</tr>
<tr>
<td>body</td>
<td>object</td>
<td>The request body of the API operation associated with this action.</td>
</tr>
</tbody>
</table>

Example:

```javascript
context.request.url;
```

**response**

Accesses the current HTTP response. For both Before and After actions, updates can be made to the response before Kibo processes it.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>object</td>
<td>The response header collection.</td>
</tr>
<tr>
<td>viewData</td>
<td>object</td>
<td>The viewData collection used by the storefront rendering engine.</td>
</tr>
<tr>
<td>viewName</td>
<td>string</td>
<td>The response viewName value.</td>
</tr>
<tr>
<td>body</td>
<td>object</td>
<td>The response body of the API operation associated with this action.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>The response status code.</td>
</tr>
<tr>
<td>message</td>
<td>string</td>
<td>The response message.</td>
</tr>
<tr>
<td>length</td>
<td>integer</td>
<td>The response length.</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>The response type.</td>
</tr>
</tbody>
</table>
Example:
context.response.header;

Context Methods Available to All HTTP Actions

request.get

Returns an HTTP header value for the specified header key.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>string</td>
<td>The header key.</td>
</tr>
</tbody>
</table>

Example:
context.request.get(field);

Response:
"object"

response.get

Gets a field from the response.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>object</td>
<td>The field to obtain.</td>
</tr>
</tbody>
</table>

Example:
context.response.get(field);

Response:
"string"

response.set

Sets the HTTP headers for the response.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>values</td>
<td>object</td>
<td>The values for the HTTP headers.</td>
</tr>
</tbody>
</table>
Example:

```
context.response.set({ myCustomHeader: "howdy" });
```

Response: N/A

**response.set2**

Updates a field in the response.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>string</td>
<td>The key of the field.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>The value of the field.</td>
</tr>
</tbody>
</table>

Example:

```
context.response.set("name", "stuff");
```

Response: N/A

**response.remove**

Removes an HTTP header from the response.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>string</td>
<td>The HTTP header to remove.</td>
</tr>
</tbody>
</table>

Example:

```
context.response.remove(header);
```

Response: N/A

**response.redirect**

Redirects the incoming URL.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>string</td>
<td>The destination URL.</td>
</tr>
</tbody>
</table>

Example:
context.response.redirect("http://someOtherSite/foo");

Response: N/A

**response.end**

Ends the response so that other actions or Kibo logic can run. Also, signals the callback to complete.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Example:

```javascript
context.response.end();
```

Response: N/A

**get.resource**

Returns the currently persisted value of the requested resource. **Not available for all calls.**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Example:

```javascript
context.get.resource();
```

Response: N/A

**get.resourceStatus**

Gets the HTTP resource status.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Example:

```javascript
context.get.resourceStatus();
```

Response: N/A

**items.urlHelper.getUrl**
Gets the current URL.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>config</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example:

context.items.urlHelper.getUrl(type, object, config);

**Context Objects Available to All Actions**

**apiContext**

Accesses tenant information.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseUrl</td>
<td>string</td>
<td>The base URL for the site.</td>
</tr>
<tr>
<td>basePciUrl</td>
<td>string</td>
<td>The base PCI URL for the site.</td>
</tr>
<tr>
<td>tenantPod</td>
<td>string</td>
<td>The name of the tenant pod in which the tenant resides.</td>
</tr>
<tr>
<td>appClaims</td>
<td>string</td>
<td>The application claims token.</td>
</tr>
<tr>
<td>appKey</td>
<td>string</td>
<td>The application key.</td>
</tr>
<tr>
<td>tenantId</td>
<td>integer</td>
<td>Unique identifier for the tenant.</td>
</tr>
<tr>
<td>siteId</td>
<td>integer</td>
<td>Unique identifier for the site. This ID is used at all levels of a store, catalog, and tenant to associate objects to a site.</td>
</tr>
<tr>
<td>masterCatalogId</td>
<td>integer</td>
<td>Unique identifier for the master catalog.</td>
</tr>
<tr>
<td>catalogId</td>
<td>integer</td>
<td>The unique identifier for the product catalog. Catalogs are part of a master catalog.</td>
</tr>
<tr>
<td>currencyCode</td>
<td>string</td>
<td>The default three-letter ISO currency code for monetary amounts.</td>
</tr>
<tr>
<td>previewDate</td>
<td>date/time</td>
<td>The date and time that the content is being viewed. This might be a future date if the content is previewed with an active date range set in the future.</td>
</tr>
<tr>
<td>Properties</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>localeCode</td>
<td>string</td>
<td>The locale code per the country code provided. This code determines the localized content to use and display.</td>
</tr>
<tr>
<td>correlationId</td>
<td>string</td>
<td>The unique identifier of the API request associated with the event action, which might contain multiple actions.</td>
</tr>
<tr>
<td>isAuthorizedAsAdmin</td>
<td>Boolean</td>
<td>Indicates whether the Dev Account user is authorized as an admin.</td>
</tr>
<tr>
<td>userClaims</td>
<td>string</td>
<td>The user claims token.</td>
</tr>
</tbody>
</table>

Example:

code: context.apiContext.baseUrl;

**configuration**

Receives a JSON response that contains information about the configuration data set in the Action Management JSON editor.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Varies</td>
<td>object</td>
<td>Custom fields and values that you can set in the Action Management JSON Editor.</td>
</tr>
</tbody>
</table>

Example:

code: context.configuration.customData;