



OCC Reference

Generated on: 2024-12-03 11:17:04 GMT+0000

SAP Commerce | 2205

Public

Original content: https://help.sap.com/docs/SAP_COMMERCE/e5d7cec9064f453b84235dc582b886da?locale=en-US&state=PRODUCTION&version=2205




Warning

This document has been generated from the SAP Help Portal and is an incomplete version of the official SAP product documentation. The information included in custom documentation may not reflect the arrangement of topics in the SAP Help Portal, and may be missing important aspects and/or correlations to other topics. For this reason, it is not for productive use.

For more information, please visit the <https://help.sap.com/docs/disclaimer>.

Omni Commerce Connect

The Omni Commerce Connect (OCC) API exposes a broad set of commerce and data services. It enables you to integrate SAP Commerce functionality anywhere in your application landscape.

Features	Architecture	Implementation
		
Enabling Interactive OCC REST API Documentation OCC Calls Security	OCC Extension Architecture OCC AddOns Architecture	OCC API Implementation OCC API v1 Reference

[OCC Extension Architecture](#)

OCC Extensions allow you to extend the functionality of OCC without the process of addon installation.

[OCC AddOns Architecture](#)

OCC is a set of extensions providing commerce-driven RESTful web services. OCC addons extend OCC with new functionality.

[OCC Calls Security](#)

The OCC calls security is ensured by highly configurable Spring security mechanisms.

[OCC API Implementation](#)

Get an overview of how the SAP Commerce OCC API is implemented, and learn what you need to know to extend it with your own custom API implementation.

[OCC API v1 Reference](#)

Version one of the OCC API offers stateful interaction with SAP Commerce functionality.

[Enabling Interactive OCC REST API Documentation](#)

OAuth clients need to be defined and authorized to enable the interactive OCC REST API documentation.

OCC API v1 Reference

Version one of the OCC API offers stateful interaction with SAP Commerce functionality.

The legacy v1 OCC API continues to be supported, but is not recommended for new development. Instead, use the current and more fully-featured v2 implementation. For more details, see [OCC API Implementation](#).

Caution

This page refers to software that has been deprecated as part of the Accelerator UI and older OCC template extensions deprecation. For more information, see [Deprecation of Accelerator UIs and Older OCC Template Extensions](#).

[OCC v1 comparison with OCC v2](#)

The SAP Commerce OCC API offers both the legacy v1 stateful implementation, alongside the newer, default stateless REST API. Get a full overview of the differences.

[OCC v1 REST Calls](#)

Get an overview of REST calls available in Version 1 of the SAP Commerce OCC API.

[OCC v1 Sample Flows](#)

Sample flow calls for the SAP Commerce OCC API in v1

[Error Responses from OCC v1](#)

Error responses returned by the `ycommercewebservices` extension.

[Cross-Origin Resource Sharing](#)

Cross-Origin Resource Sharing (CORS) is a W3C effort to introduce a standard mechanism for enabling cross-domain requests in web browsers and participating servers.

[Upgrading to OCC Extensions](#)

Guidelines on upgrading from v1 to Extensions-based implementation of the OCC API.

OCC v1 comparison with OCC v2

The SAP Commerce OCC API offers both the legacy v1 stateful implementation, alongside the newer, default stateless REST API. Get a full overview of the differences.

Overview

The SAP Commerce OCC API has two available implementations. OCC v2 ensures a configurable and stateless implementation of RESTful web services, supporting future growth in a headless commerce environment. However, the legacy v1 implementation is still supported for existing customers using that version. For documentation on the default version two, see [OCC API Implementation](#).

Version 2 (default)

The key features of Version 2 are as follows:

Feature	Description
Stateless	<ul style="list-style-type: none">• The calls are now stateless so the customer data is no longer preserved between subsequent requests.• Each time the customer needs to provide all the required data such as customer id or cart id.
RESTful implementation	<ul style="list-style-type: none">• URL resources have been refactored and are now more RESTful.• The customer needs to provide resource type and resource identifier of the resource he would like to work with.
Data creation	<ul style="list-style-type: none">• Executed using the URL parameter list or RequestBody.

Version 1

The key features of Version 1 are as follows:

Feature	Description
Stateful	<ul style="list-style-type: none">• A stateful way of communication with commerce layer• Flows some extent similar to storefront flows, based on the <code>accelerator-services</code> extension (for example the checkout process).• The customer cart and other customer data are stored in the session, and are preserved between subsequent requests (basing on the assumption the JSESSIONID is stored).• The customer can be more focused on the actual actions he wants to perform and not on holding and providing the whole context data.
Response format	<ul style="list-style-type: none">• Responses are provided in XML or JSON format., depending on the request.

Version Resources Separation

Separate Servlets

For each version, there is a separate servlet defined in the `web.xml` file. These servlets have different java-based configurations. There is also a servlet defined for OAuth authorization - in this case it is a feature common for both versions.

v1

The v1 servlet definition is as follows:

```
...
<servlet>
<servlet-name>springmvc-v1</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<init-param>
  <param-name>contextClass</param-name>
  <param-value>
    org.springframework.web.context.support.AnnotationConfigWebApplicationContext
  </param-value>
</init-param>
<init-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    de.hybris.platform.ycommercewebservices.v1.config.WebConfig
  </param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>springmvc-v1</servlet-name>
  <url-pattern>/v1/*</url-pattern>
</servlet-mapping>
...
```

v2

The v2 servlet definition is as follows:

```
...
<servlet>
  <servlet-name>springmvc-v2</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextClass</param-name>
    <param-value>
      org.springframework.web.context.support.AnnotationConfigWebApplicationContext
    </param-value>
  </init-param>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
      de.hybris.platform.ycommercewebservices.v2.config.WebConfig
    </param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>springmvc-v2</servlet-name>
  <url-pattern>/v2/*</url-pattern>
</servlet-mapping>
...
```

Spring Configuration

Separate Web Context for Servlets

Each version has its own web context defined based on the Java configuration. The Java configuration defines the request mapping handler, message converters, and handler exception resolvers. It also imports standard XML configuration for the proper version.

WebConfig.java configuration for v1:

```

/**
 * Spring configuration which replace <mvc:annotation-driven> tag. It allows override default
 * RequestMappingHandlerMapping with our own mapping handler
 *
 */
@Configuration
@ImportResource(
{ "WEB-INF/config/v1/springmvc-v1-servlet.xml" })
public class WebConfig extends WebMvcConfigurationSupport
{
    @Resource
    private List<HttpMessageConverter<?>> messageConvertersV1;

    @Resource
    private List<HandlerExceptionResolver> exceptionResolversV1;

    private ApplicationContext applicationContext;

    @Override
    @Bean
    public RequestMappingHandlerMapping requestMappingHandlerMapping()
    {
        final CommerceHandlerMapping handlerMapping = new CommerceHandlerMapping("v1");
        handlerMapping.setOrder(0);
        handlerMapping.setDetectHandlerMethodsInAncestorContexts(true);
        handlerMapping.setInterceptors(getInterceptors());
        handlerMapping.setContentNegotiationManager(mvcContentNegotiationManager());
        return handlerMapping;
    }

    @Override
    protected void configureMessageConverters(final List<HttpMessageConverter<?>> converters)
    {
        converters.addAll(messageConvertersV1);
    }

    @Override
    protected void configureHandlerExceptionResolvers(final List<HandlerExceptionResolver> exceptionResolvers)
    {
        final ExceptionHandlerExceptionResolver exceptionHandlerExceptionResolver = new ExceptionHandlerExceptionRe
        exceptionHandlerExceptionResolver.setApplicationContext(applicationContext);
        exceptionHandlerExceptionResolver.setContentNegotiationManager(mvcContentNegotiationManager());
        exceptionHandlerExceptionResolver.setMessageConverters(getMessageConverters());
        exceptionHandlerExceptionResolver.afterPropertiesSet();

        exceptionResolvers.add(exceptionHandlerExceptionResolver);
        exceptionResolvers.addAll(exceptionResolversV1);
        exceptionResolvers.add(new ResponseStatusExceptionHandler());
        exceptionResolvers.add(new DefaultHandlerExceptionHandler());
    }

    @Override
    public void setApplicationContext(final ApplicationContext applicationContext) throws BeansException
    {
        super.setApplicationContext(applicationContext);
        this.applicationContext = applicationContext;
    }
}

```

WebConfig.java configuration for v2:

```

/**
 * Spring configuration which replace <mvc:annotation-driven> tag. It allows override default
 * RequestMappingHandlerMapping with our own mapping handler
 *
 */
@Configuration
@ImportResource(
{ "WEB-INF/config/v2/springmvc-v2-servlet.xml" })
public class WebConfig extends WebMvcConfigurationSupport
{
    @Resource
    private List<HttpMessageConverter<?>> messageConvertersV2;

    @Resource
    private List<HandlerExceptionResolver> exceptionResolversV2;

    private ApplicationContext applicationContext;

    @Override
    @Bean
    public RequestMappingHandlerMapping requestMappingHandlerMapping()
    {
        final CommerceHandlerMapping handlerMapping = new CommerceHandlerMapping("v2");
        handlerMapping.setOrder(0);
        handlerMapping.setDetectHandlerMethodsInAncestorContexts(true);
        handlerMapping.setInterceptors(getInterceptors());
        handlerMapping.setContentNegotiationManager(mvcContentNegotiationManager());
        return handlerMapping;
    }

    @Override
    protected void configureMessageConverters(final List<HttpMessageConverter<?>> converters)
    {
        converters.addAll(messageConvertersV2);
    }

    @Override
    protected void configureHandlerExceptionResolvers(final List<HandlerExceptionResolver> exceptionResolvers)
    {
        final ExceptionHandlerExceptionResolver exceptionHandlerExceptionResolver = new ExceptionHandlerExceptionRe
        exceptionHandlerExceptionResolver.setApplicationContext(applicationContext);
        exceptionHandlerExceptionResolver.setContentNegotiationManager(mvcContentNegotiationManager());
        exceptionHandlerExceptionResolver.setMessageConverters(getMessageConverters());
        exceptionHandlerExceptionResolver.afterPropertiesSet();

        exceptionResolvers.add(exceptionHandlerExceptionResolver);
        exceptionResolvers.addAll(exceptionResolversV2);
        exceptionResolvers.add(new ResponseStatusExceptionHandler());
        exceptionResolvers.add(new DefaultHandlerExceptionHandler());
    }

    @Override
    public void setApplicationContext(final ApplicationContext applicationContext) throws BeansException
    {
        super.setApplicationContext(applicationContext);
        this.applicationContext = applicationContext;
    }
}

```

Common Context

In the `web.xml` file there is a common context defined. Beans defined in this context are available for both servlets. This context should contain security configuration and beans used in filters.

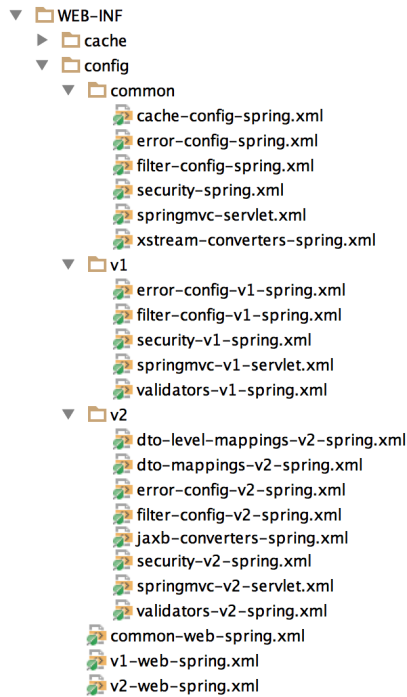
```

...
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>WEB-INF/ycommercewebservices-web-spring.xml</param-value>
    </context-param>
...

```

Configuration Files Localization

Spring configuration files can be found in the `WEB-INF/config` directory. Configuration files specific for versions are in `v1`, `v2` sub-directories.



Enabling Version 1

The old v1 version is turned off by default. To enable it, uncomment the filters and the servlet definition in the `web.xml` file as shown in the example below.

```

<!-- Uncomment to make v1 version available -->
<!--
<filter>
    <description>
        Spring configured chain of spring filter beans
    </description>
    <filter-name>commerceWebServicesFilterChainV1</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
    <filter-name>commerceWebServicesFilterChainV1</filter-name>
    <servlet-name>springmvc-v1</servlet-name>
</filter-mapping>
<filter-mapping>
    <filter-name>httpPutFormFilter</filter-name>
    <servlet-name>springmvc-v1</servlet-name>
</filter-mapping>
<servlet>
    <servlet-name>springmvc-v1</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextClass</param-name>
        <param-value>
            org.springframework.web.context.support.AnnotationConfigWebApplicationContext
        </param-value>
    </init-param>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            de.hybris.platform.ycommercewebservices.v1.config.WebConfig
        </param-value>
    </init-param>
</servlet>
-->

```

```

        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc-v1</servlet-name>
        <url-pattern>/v1/*</url-pattern>
    </servlet-mapping>
    -->
    <!-- END Uncomment to make v1 version available -->

```

Additionally, enable v1 spring configuration in the `ycommercewebservices-web-spring.xml` file:

```

<!-- Uncomment to enable version v1 -->
<!--<import resource="config/v1-web-spring.xml"/>-->

```

Related Information

[RESTful Implementation](#)

[Upgrading to OCC Extensions](#)

OCC v1 REST Calls

Get an overview of REST calls available in Version 1 of the SAP Commerce OCC API.

Caution

This page refers to software that has been deprecated as part of the Accelerator UI and older OCC template extensions deprecation. For more information, see [Deprecation of Accelerator UIs and Older OCC Template Extensions](#).

Overview

Currently it is possible to retrieve a list of all products fulfilling given criteria and details of a product specified by its code. The child pages of this document hold information about currently available calls grouped by resource. Each document contains tables presenting examples of the URL requests along with available parameters.

Localization Request Parameters

Each of the calls can contain additional URL parameters that change the localization of the returned objects. The common parameters are:

- `<lang>`: Changes the language of the localized values in the response. Provide the language ISO code as a value. If no `<lang>` parameter is provided, then the response is localized with the default language of your base store.
- `<curr>`: Changes the currency of your web service call. This means that all the calculations are performed in the requested currency and all the price values are presented using the requested currency. Provide currency ISO code as a value. If no `<curr>` parameter is provided, then the default currency of your base store is used.

You can use these parameters with every requested resource. Parameters are handled separately in a specialized HTTP request filter. Check the following examples:

- `https://localhost:9001/rest/v1/mysite/cart?curr=USD&lang=en`: Use English language and US Dollar for the request.
- `https://localhost:9001/rest/v1/mysite/cart/entry?curr=USD`: Use US Dollar and default language of mysite's store.
- `https://localhost:9002/rest/v1/mysite/customers/current?lang=de`: Use German language and default currency of mysite's store.

Site Parameters

In many URLs of resources there is a site path parameter, for example `https://localhost:9001/rest/v1/electronics/catalogs` where `<electronics>` is the `<BaseSite.uid>` property of the website. In your custom implementation you need to add your own base site:

- URL pattern, for example `(?i)^https?://localhost(:[0-9]+)?/rest/.*$`

Configure `productUrlPattern`, for example `{category-path}/{product-name}/p/{product-code}`

You can use following parameters:

- `<{baseSite-uid}>`
- `<{category-path}>`
- `<{product-name}>`
- `<{product-code}>`
- Configure `categoryUrlPattern`, for example `/ {category-path} /c/{category-code}`

You can use following parameters:

- `<{baseSite-uid}>`
- `<{category-path}>`
- `<{category-code}>`
- `<{catalog-id}>`
- `<{catalogVersion}>`

Here is an example of the response from:

`https://localhost:9001/rest/v1/electronics/catalogs/electronicsProductCatalog/Online/categories/brands?options=PRODUCTS`

```
<category>
  <id>brands</id>
  <lastModified>2012-08-13T09:11:00+02:00</lastModified>
  <name>Brands</name>
  <url>/Brands/c/brands</url>

<subcategories/>
  <products>
    <product>
      <averageRating>3.5</averageRating>
      <purchasable>true</purchasable>
      <code>478828</code>
      <url>/Open-Catalogue/Cameras/Digital-Cameras/Digital-SLR/10-2-Megapixel-D-SLR-with-Standard-Zoom-Lens/p/478828</url>
      <manufacturer>Sony</manufacturer>
      <name>10.2 Megapixel D-SLR with Standard Zoom Lens</name>
    </product>
  </products>
</subcategories>
```

If you do not specify the site path parameter or it does not exist, you will receive the following exception.

```
de.hybris.platform.ycommercewebservices.exceptions.InvalidResourceException: Base site electronis doesn't exist
```

You can easily configure the site properties using the Backoffice Administration Cockpit under  [WCMS](#)  [Website](#) .

Stateless and Stateful Usage of the API

While most of the API resources can be used in a stateless fashion, there are some resources that require the API client to keep track of cookies for the session management. Generally, we advise the API clients to keep track of all cookies they receive from the server and automatically overwrite existing cookies in case the server sends a new cookie value for an existing cookie name.

Using cookies for session management is especially important for all **Cart Resources**. A POST to `/ {site} /cart/entry` would create a new in-memory cart each time you execute the request if you omit the cookies that you receive from the server. If you omit passing the cookies, you can neither keep the cart of the customer, nor add multiple products in separate requests that way.

New cookies are sent from the server by HTTP responses and the Set-Cookie header. Your API client needs to check if a Set-Cookie header is available. If it is available, then extract the cookie information. See an example based on Groovy code.

```
//con is a HttpURLConnection or a HttpsURLConnection
def cookie = con.getHeaderField('Set-Cookie') //for example JSESSIONID=C69059FD51C68E610321A818B0019DB2; Path=/rest;
def cookieNoPath = cookie.split(';')[0] //only the cookie value now
```

Once your API issues an HTTP request to call one of the resources, the cookie needs to be a part of the request headers.

```
con.setRequestProperty("Cookie", cookie)
```

Resource Cart

Overview of the available **Cart** resources.

i Note

The URL examples are used for presentation purposes only. In a customized environment, you must replace the server address with your own.

Sequence of Calling Cart API Methods

The sequence of calling methods from the Cart resource is essential. The following table presents an example for a simple workflow.

Method	URL	Request Body
POST	https://localhost:9002/rest/v1/electronics/customers	
POST	https://localhost:9002/authorizationserver/oauth/token	
POST	https://localhost:9002/rest/v1/customers/current/addresses	
GET	https://localhost:9002/rest/v1/electronics/cart	
POST	https://localhost:9002/rest/v1/electronics/cart/entry	
POST	https://localhost:9002/rest/v1/electronics/cart/entry	
GET	https://localhost:9002/rest/v1/electronics/cart	
PUT	https://localhost:9002/rest/v1/electronics/cart/entry/0	
DELETE	https://localhost:9002/rest/v1/electronics/cart/entry/0	
GET	https://localhost:9002/rest/v1/electronics/cart	
PUT	https://localhost:9002/rest/v1/electronics/cart/address/delivery/\$id\$	

Method	URL	Request Body
DELETE	https://localhost:9002/rest/v1/electronics/cart/address/delivery	
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/address/delivery/\$id\$	
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/premium-gross	
DELETE	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/standard-gross	
GET	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos	
PUT	https://localhost:9002/rest/v1/electronics/cart/paymentinfo/\$paymentInfoId\$	
POST	https://localhost:9002/rest/v1/electronics/cart/authorize	
POST	https://localhost:9002/rest/v1/electronics/cart/placeorder	
GET	https://localhost:9002/rest/v1/orders	
GET	https://localhost:9002/rest/v1/orders/\$orderNumber\$	
POST	https://localhost:9002/rest/v1/customers/current/logout	
POST	https://localhost:9002/rest/v1/electronics/cart/paymentinfo	Should set header Content-Type: to application/x-www-form-urlencoded

Method	URL	Request Body
		<pre> accountHolderName: "Anja Hertz" cardNumber: "4111111111111111" cardType: "visa" expiryMonth: "01" expiryYear: "2013" saved: "true" defaultPaymentInfo: "true" billingAddress.titleCode: "mr" billingAddress.firstName: "Anja" billingAddress.lastName: "Hertz" billingAddress.line1: "test1" billingAddress.line2: "test2" billingAddress.postalCode: "12345" billingAddress.town: "somecity" billingAddress.country.isocode: ' </pre>

Resource: /{site}/cart


Method	GET
Description	<p>Returns a session cart if such a cart exists. If there was no cart in the current session:</p> <ul style="list-style-type: none"> For a logged in user, cart is restored if it exists, and returned For anonymous user or when there is no saved cart, an empty mock-cart data object is created and returned
Example URL	https://localhost:9002/rest/v1/electronics/cart
Authentication	None
Request Parameters	Not applicable
Headers	<ul style="list-style-type: none"> Accept <p>(Style: header, Required: true):</p> <p>Accept header needs to be sent with each request. It can be application/xml or application/json</p>
Representations	application/xml, application/json
Cookies	It requires session information sent in a cookie or via ;sessionId=... in the URL. Otherwise a new session and therefore new cart is created.

Resource: /{site}/cart/entry

Method	POST
Description	Adds a product to the session cart. If session cart does not exist yet, it is created beforehand.
Example URL	https://localhost:9002/rest/v1/electronics/cart/entry
Authentication	None
Request Parameters	<p>Parameters need to be provided as POST body.</p> <ul style="list-style-type: none"> <code> <p>(Style: code, Required: true):</p> <p>Code of the product to be added to cart. Product look-up is performed for the current session product catalog version.</p>

	<ul style="list-style-type: none"> • <qty> (Style: qty, Required: false, Default: 1): New cart item's quantity.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json
Representations	application/xml, application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Otherwise new session and therefore new cart is created. For the cart to be accessible in next requests, add JSESSIONID cookie to the next calls.

Resource: /{site}/cart/entry/{entryNumber}

Method	PUT
Description	Updates the cart entry quantity. If there is no cart entry with the requested number, the CommerceCartModificationException is returned in the response.
Example URL	https://localhost:9002/rest/v1/electronics/cart/entry/0?qty=5
Authentication	None
Request Parameters	Parameters need to be provided in the PUT body. HttpPutFormContentFilter  is used to read the PUT body. <ul style="list-style-type: none"> • <qty> (Style: qty, Required: true): New quantity for cart entry.
Headers	<ul style="list-style-type: none"> • Content-Type application/x-www-form-urlencoded (Style: header, Required: true): The application/x-www-form-urlencoded content type is required by HttpPutFormContentFilter • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json
Representations	application/xml, application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Otherwise new session and therefore new cart is created. For the cart to be accessible in next requests, add JSESSIONID cookie to the next calls.

Resource: /{site}/cart/entry/{entryNumber}

Method	DELETE
Description	Deletes cart entry. If there is no cart entry with the requested number, the CommerceCartModificationException is returned in the response.

Example URL	https://localhost:9002/rest/v1/electronics/cart/entry/0
Authentication	None
Request Parameters	Not applicable.
Headers	<ul style="list-style-type: none"> • Accept <p>(Style: header, Required: true):</p> <p>Accept header needs to be sent with each request. It can be application/xml or application/json</p>
Representations	application/xml, application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Otherwise a new session and therefore a new cart is created. For the cart to be accessible in next requests, add JSESSIONID cookie to the next calls.

Resource: /{site}/cart/address/delivery/{id}

Method	PUT
Description	Sets a delivery address for a session cart. The address ID must be a PK of a current address. The address country must be among the delivery countries of a current base store. Method responses with cart data that should reflect the change in the cart.
Checkout State Prerequisites	There must be a session cart created.
Example URL	https://localhost:9002/rest/v1/electronics/cart/address/delivery/123
Authentication	Access is restricted to the ROLE_CUSTOMERGROUP . This method is restricted to HTTPS channel.
Request Parameters	Not applicable.
Headers	<ul style="list-style-type: none"> • Accept <p>(Style: header, Required: true):</p> <p>Accept header needs to be sent with each request. It can be application/xml or application/json</p>
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Otherwise a new session and subsequently a new cart is created. For the cart to be accessible in next requests, add JSESSIONID cookie to the next calls.

Resource: /{site}/cart/address/delivery/

Method	DELETE
Description	Deletes delivery address from a session cart. The response includes cart data that should reflect the change in the cart.
Example URL	https://localhost:9002/rest/v1/electronics/cart/address/delivery
Authentication	Access is restricted to the ROLE_CUSTOMERGROUP . This method is restricted to HTTPS channel.
Request Parameters	Not applicable.
Headers	<ul style="list-style-type: none"> • Accept <p>(Style: header, Required: true):</p>

	Accept header needs to be sent with each request. It can be application/xml or application/json
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Otherwise a new session and subsequently a new cart is created. For the cart to be accessible in next requests, add JSESSIONID cookie to the next calls.

Resource: /{site}/cart/deliverymodes/

Method	GET
Description	Returns all delivery modes supported for the current base store and cart delivery address.
Checkout State Prerequisites	There must be a delivery address for the cart, otherwise an empty list will be returned.
Example URL	https://localhost:9002/rest/v1/electronics/cart/deliverymodes
Authentication	Access is restricted to the ROLE_CUSTOMERGROUP . This method is restricted to HTTPS channel.
Request Parameters	Not applicable.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Otherwise a new session and subsequently a new cart is created. For the cart to be accessible in next requests, add JSESSIONID cookie to the next calls

Resource: /{site}/cart/deliverymodes/{code}

Method	PUT
Description	Sets delivery mode according to the code for the current cart. Method responses with the cart data that should reflect the change in the cart.
Example URL	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/standard-gross
Authentication	Access is restricted to the ROLE_CUSTOMERGROUP . This method requires basic HTTP authentication and is restricted to HTTPS channel.
Request Parameters	Not applicable.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json .
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Otherwise a new session and subsequently a new cart is created. For the cart to be accessible in the next requests, add JSESSIONID cookie to the next calls.

Resource: /{site}/cart/deliverymodes/

Method	DELETE
Description	Removes delivery mode from the current cart. The response includes cart data that should reflect the change in the cart.
Example URL	https://localhost:9002/rest/v1/electronics/cart/deliverymodes
Authentication	Access is restricted to the ROLE_CUSTOMERGROUP . This method requires basic HTTP authentication and is restricted to HTTPS channel.
Request Parameters	Not applicable.
Headers	<ul style="list-style-type: none"> • Accept <p>(Style: header, Required: true):</p> <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via ;sessionId=... in the URL. Otherwise a new session and subsequently a new cart is created. For the cart to be accessible in next requests, add JSESSIONID cookie to the next calls.

Resource: /{site}/cart/paymentinfo/

Method	POST
Description	Creates a new credit card payment info for the current user. The response includes the payment info data.
Checkout State Prerequisites	There must be a session cart.
Example URL	https://localhost:9002/rest/v1/electronics/cart/paymentinfo
Authentication	Access is restricted to the ROLE_CUSTOMERGROUP . This method requires basic HTTP authentication and is restricted to HTTPS channel.
Request Body	<p>Need to be provided as URL encoded post body:</p> <ul style="list-style-type: none"> • <accountHolderName> <p>(Required: true):</p> <ul style="list-style-type: none"> • <cardNumber> <p>(Required: true):</p> <ul style="list-style-type: none"> • <cardType> <p>(Required: true):</p> <p>Call GET /{site}/cardtypes beforehand to see what card types are supported</p> <ul style="list-style-type: none"> • <expiryMonth> <p>(Required: true):</p> <ul style="list-style-type: none"> • <expiryYear> <p>(Required: true):</p> <ul style="list-style-type: none"> • <saved> <p>(Required: false):</p> <p>If the payment info should be saved for the customer and than could be reused for future orders.</p>

	<ul style="list-style-type: none"> • <code><defaultPaymentInfo></code> (Required: false, Default: false): • <code><billingAddress.titleCode></code> (Required: true): • <code><billingAddress.firstName></code> (Required: true): • <code><billingAddress.lastName></code> (Required: true): • <code><billingAddress.line1></code> (Required: true): • <code><billingAddress.line2></code> (Required: false): • <code><billingAddress.postalCode></code> (Required: true): • <code><billingAddress.town></code> (Required: true): • <code><billingAddress.country.isocode></code> (Required: true):
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Otherwise a new session and subsequently a new cart is created. For the cart to be accessible in next requests, add JSESSIONID cookie to the next calls.

Resource: `/[site]/cart/paymentinfo/{id}`

Method	PUT
Description	Adds a credit card payment info according to the ID of the current user's cart.
Checkout State Prerequisites	There must be a session cart. The response includes cart data that should reflect the change in the cart.
Example URL	<code>https://localhost:9002/rest/v1/electronics/cart/paymentinfo/1234</code>
Authentication	Access is restricted to the ROLE_CUSTOMERGROUP . This method requires basic HTTP authentication and is restricted to HTTPS channel.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Otherwise a new session and subsequently a new cart is created. For the cart to be accessible in

Resource: /{site}/cart/authorize

Method	POST
Description	Authorizes the credit card payment with the CCV security code. The response contains the cart data.
Checkout State Prerequisites	There must be a session cart with associated credit card payment. The response contains cart data, that should reflect the change in the cart.
Example URL	https://localhost:9002/rest/v1/electronics/cart/authorize
Authentication	Access is restricted to the ROLE_CUSTOMERGROUP . This method requires basic HTTP authentication and is restricted to HTTPS channel.
Request Parameters	Need to be provided as URL encoded post body: <ul style="list-style-type: none"> • <securityCode> (Required: true):
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Otherwise a new session and subsequently a new cart is created. For the cart to be accessible in next requests, add JSESSIONID cookie to the next calls.

Resource: /{site}/cart/placeorder

Method	POST
Description	Places an order based on the session cart. The response contains the new order data.
Checkout State Prerequisites	User session must have a valid cart.
Example URL	https://localhost:9002/rest/v1/electronics/cart/placeorder
Authentication	Access is restricted to the ROLE_CUSTOMERGROUP . This method requires basic HTTP authentication and is restricted to HTTPS channel.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json .
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Use JSESSIONID cookie to the next calls.

Resource: /{site}/cart/restore

Method	GET
Description	Restores anonymous cart by guid .
Example URL	https://localhost:9002/rest/v1/electronics/cart/restore?guid=xxxxx
Authentication	Access is restricted to the ROLE_CLIENT and ROLE_TRUSTED_CLIENT . This method is restricted to HTTPS channel.
Headers	<ul style="list-style-type: none"> • Accept <p>(Style: header, Required: true):</p> <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Representations	application/xml , application/json

Resource: /{site}/cart/promotion/{promotionCode}

Method	POST
Description	Enables order promotion given by promotionCode for a current cart. If promotion is applied successfully, then the modified cart will be returned.
Example URL	https://localhost:9002/rest/v1/electronics/cart/OrderThreshold15Discount
Authentication	Access is restricted to the ROLE_CLIENT , ROLE_CUSTOMERGROUP , and ROLE_TRUSTED_CLIENT . This method is restricted to HTTPS channel.
Headers	<ul style="list-style-type: none"> • Accept <p>(Style: header, Required: true):</p> <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Use JSESSIONID cookie to the next calls.

Resource: /{site}/cart/promotion/{promotionCode}

Method	DELETE
Description	Disables order promotion given by promotionCode for a current cart. If promotion is removed successfully, then the modified cart will be returned.
Example URL	https://localhost:9002/rest/v1/electronics/cart/promotion/OrderThreshold15Discount
Authentication	Access is restricted to the ROLE_CLIENT , ROLE_CUSTOMERGROUP and TRUSTED_CLIENT . This method requires client or customer HTTP authentication and is restricted to HTTPS channel.
Headers	<ul style="list-style-type: none"> • Accept <p>(Style: header, Required: true):</p> <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via ;jsessionid=... in the URL. Use JSESSIONID cookie to the next calls.

Resource: `/[site]/cart/voucher/{voucherCode}`

Method	POST
Description	Applies a voucher given by a voucherCode for the current cart. If the voucher code is applied successfully, then the modified cart will be returned. The JSON response should contain the appliedVouchers attribute with the voucher information. The XML response should contain the appliedVouchers tag with the voucher information.
Example URL	<code>https://localhost:9002/rest/v1/electronics/cart/voucher/xyz-MHE2-B8L5-LPHE</code>
Authentication	Access is restricted to the ROLE_CLIENT , ROLE_CUSTOMERGROUP and ROLE_TRUSTED_CLIENT . This method is restricted to HTTPS channel.
Request Parameters	Not applicable.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via <code>jsessionid=...</code> in the URL. Use JSESSIONID cookie to the next calls.

Resource: `/[site]/cart/voucher/{voucherCode}`

Method	DELETE
Description	<p>Method removes a voucher given by the voucherCode from the current cart. If the voucher code is removed successfully, then the modified cart will be returned.</p> <ul style="list-style-type: none"> • In JSON response, the appliedVouchers attribute should not contain released voucher anymore. <code>"appliedVouchers": []</code> • In XML response, the appliedVouchers tag should not contain released voucher anymore. <code><appliedVouchers /></code>
Example URL	<code>https://localhost:9002/rest/v1/electronics/cart/voucher/xyz-MHE2-B8L5-LPHE</code>
Authentication	Access is restricted to the ROLE_CLIENT , ROLE_CUSTOMERGROUP and TRUSTED_CLIENT . This method is restricted to HTTPS channel.
Request Parameters	Not applicable.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml , application/json
Cookies	It requires session information sent in a cookie or via <code><jsessionid=...></code> in the URL. Use JSESSIONID cookie to the next calls.

Resource Catalogs

an overview on the available Catalogs resource

i Note

The URL examples are used for presentation purposes only. In a customized environment, you must replace the server address with your own.

Resource: /{site}/catalogs

Method	GET
Description	Returns all catalogs with versions that are defined for the base store.
Example URL	https://localhost:9002/rest/v1/electronics/catalogs
Request Parameters	<ul style="list-style-type: none"> • <i><options></i> (Style: query, Required: false): <p>It defines the level of details requested. This parameter can have a combination of the following values, separated by .: BASIC, CATEGORIES, PRODUCTS, SUBCATEGORIES. Keep in mind that the value needs to be properly URL-encoded.</p> <p>Options are dependent on each other: for example PRODUCTS option makes sense only if CATEGORIES option is also used, as the products elements are embedded in categories in the response object.</p> <p>Example: <i><options>=CATEGORIES,PRODUCTS</i> requests root categories with products.</p>
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Representations	application/xml, application/json

Resource: /{site}/catalogs/{id}

Method	GET
Description	Returns catalog by ID with versions defined for the current base store.
Example URL	https://localhost:9002/rest/v1/electronics/catalogs/electronicsProductCatalog
Request Parameters	<ul style="list-style-type: none"> • <i><options></i> (Style: query, Required: false): <p>It defines the level of details requested. This parameter can have a combination of the following values, separated by .: BASIC, CATEGORIES, PRODUCTS, SUBCATEGORIES. Keep in mind that the value needs to be properly URL-encoded.</p> <p>Options are dependent on each other: PRODUCTS option makes sense only if CATEGORIES option is also used, as products elements are embedded in categories in the response object.</p> <p>Example: <i><options>=CATEGORIES, PRODUCTS</i> requests root categories with products.</p>
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true):

	Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: /{site}/catalogs/{catalogId}/{catalogVersionId}

Method	GET
Description	Returns information about catalog version that exists for the current base store.
Example URL	https://localhost: 9002/rest/v1/electronics/catalogs/electronicsProductCatalog/Online
Request Parameters	<ul style="list-style-type: none"> <options> (Style: query, Required: false): <p>It defines the level of details requested. This parameter can have a combination of the following values, separated by.:BASIC,CATEGORIES,PRODUCTS,SUBCATEGORIES. Keep in mind that the value needs to be properly URL-encoded.</p> <p>Options are dependent on each other: for example PRODUCTS option makes sense only if CATEGORIES option is also used, as products elements are embedded in categories in the response object.</p> <p>Example: <options>=CATEGORIES,PRODUCTS requests root categories with products.</p>
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Representations	application/xml, application/json

Resource: /{site}/catalogs/{catalogId}/{catalogVersionId}/categories/{categoryCode}

Method	GET
Description	Returns information about category that exist in a base store's catalog version.
Example URL	https://localhost: 9002/rest/v1/electronics/catalogs/electronicsProductCatalog/Online/categories/1
Request Parameters	<ul style="list-style-type: none"> <options> (Style: query, Required: false): <p>It defines the level of details requested. This parameter can have a combination of the following values, separated by.:BASIC,PRODUCTS,SUBCATEGORIES. Keep in mind that the value needs to be properly URL-encoded.</p> <p>Options are dependent on each other: PRODUCTS option makes sense only if CATEGORIES option is also used, as products elements are embedded in categories in the response object.</p> <p>Example: <options>=CATEGORIES,PRODUCTS requests root categories with products.</p>
Headers	<ul style="list-style-type: none"> Accept(Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Representations	application/xml, application/json

Resource Customergroup

An overview of the available **customergroup** resources.

i Note

The URL examples are used for presentation purposes only. In a customized environment, you must replace the server address with your own.

Resource: /{site}/customergroups

Method	POST
Description	Creates new customer group - direct subgroup of a customergroup . The method requires ROLE_CUSTOMERMANGAGERGROUP and secured HTTPS channel.
Example URL	https://localhost:9002/rest/v1/electronics/customergroups
Request Parameters	<ul style="list-style-type: none"> <uid> (Style: body, Required: true): Uid for new customer group. <localizedName >(Style: body, Required: false): Name in current locale.
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: /{site}/customergroups/{uid}/members

Method	PUT
Description	Assign users to the customer group with given <uid> . Requires CUSTOMERMANGAGERGROUP role and HTTPS secured channel.
Example URL	https://localhost:9002/rest/v1/electronics/customergroups/customergroup/members
Request Parameters	<ul style="list-style-type: none"> <member> (Style: body, Required: true): List of users' uid's to assign to customer group. List should be in form: <member>=<uid1>&<member>=<uid2>...
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: /{site}/customergroups

Method	GET
Description	Returns all customer groups that are direct subgroups of customergroup . Requires CUSTOMERMANGAGERGROUP role and HTTPS secure channel.

Example URL	<code>https://localhost:9002/rest/v1/electronics/customergroups?currentPage=0&pageSize=20</code>
Request Parameters	<ul style="list-style-type: none"> <code><currentPage></code> (Style: url parameter, Required: false, Default: 0): current page number(starts with 0) <code><pageSize></code> (Style: url parameter, Required: false, Default: 2147483647): number of customer group returned in one page
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: `/[site]/customergroups/{uid}`

Method	GET
Validity	5.0
Description	Returns customer group with specified uid . Requires CUSTOMERMANAGERGROUP role. The requested group must be a sub group of customergroup , otherwise InvalidCustomerGroupException is thrown. Requires secure channel.
Example URL	<code>https://localhost:9002/rest/v1/electronics/customergroups/customergroup/{myGroup}?options=MEMBERS,SUBGROUPS</code>
Request Parameters	<ul style="list-style-type: none"> <code><options></code> (Style: url parameter, Required: false): BASIC - default value. Response contains only <code><uid></code>, <code><localized name></code> and number of members, MEMBERS - with this option, response will contain information about member users (customers), SUBGROUP - with this option the response will contain information about this group subgroups. You can combine options.
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: `/[site]/customergroups/{uid}/members/{userId}`

Method	DELETE
Description	Remove user with given <code><userId></code> from customer group with given <code><uid></code> . Requires CUSTOMERMANAGERGROUP role.
Example URL	<code>https://localhost:9002/rest/v1/electronics/customergroups/customergroup/members/dab61415-7832-4daf-ae1d-c48988e5bfef</code>
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.

Representations	application/xml, application/json
-----------------	-----------------------------------

Resource Customers

An overview on the available **Customers** resources.

i Note

The URL examples are used for presentation purposes only. In a customized environment, you must replace the server address with your own.

Resource: /customers/current/logout

Method	POST
Description	Web service to logs out the customer. Terminates session. Incorporates spring security logout filter.
Example URL	https://localhost:9002/rest/v1/customers/current/logout
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>

Resource: /{site}/customers

Method	POST
Description	Register a customer.
Example URL	<ul style="list-style-type: none"> Access is restricted to ROLE_CLIENT so that only registered API clients can gain it. The client access token must be sent with the request. To obtain the token for a given <i><client_id></i> an authorization call must be sent first: <ul style="list-style-type: none"> curl --location 'https://localhost:9002/authorizationserver/oauth' \ --header 'Content-Type: application/x-www-form-urlencoded' \ --data-urlencode 'client_secret={clientSecret\}' \ --data-urlencode 'grant_type=client_credentials' \ --data-urlencode 'client_id={client_id}' Now you can send a registration call: <ul style="list-style-type: none"> https://localhost:9002/rest/v1/electronics/customers
Request Parameters	<ul style="list-style-type: none"> <i><login></i> (Style: body, Required: true): Customer's login. Customer login is case insensitive. <i><password></i> (Style: body, Required: true): Customer's password. <i><firstName></i> (Style: body, Required: true): Customer's first name. <i><lastName></i> (Style: body, Required: true): Customer's last name. <i><titleCode></i> (Style: body, Required: false): Customer's title code. For a list of codes, see StoreController

	<ul style="list-style-type: none"> • <code><access_token></code> (Style: body, Required: false): Access token granted for the given client.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: `/[site]/customers/current/addresses/default/{id}`

Method	PUT
Description	Set address as customer's default address.
Example URL	<code>https://localhost:9002/rest/v1/electronics/customers/current/addresses/default/12345</code>
Request Parameters	<ul style="list-style-type: none"> • <code><id></code> (Style: body, Required: true): Address id.
Authentication	<ul style="list-style-type: none"> • Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> o <code>curl --location 'https://localhost:9002/authorizationserver/oauth' \</code> <code>--header 'Content-Type: application/x-www-form-urlencoded' \</code> <code>--data-urlencode 'username={uid}' \</code> <code>--data-urlencode 'password={pwd}' \</code> <code>--data-urlencode 'client_secret={clientSecret\}' \</code> <code>--data-urlencode 'grant_type=password' \</code> <code>--data-urlencode 'client_id={client_id}'</code> • HTTPS channel is required
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: `/[site]/customers/current/profile`

Method	POST
Description	Update customer's profile.
Example URL	<code>https://localhost:9002/rest/v1/electronics/customers/current/profile</code>
Request Parameters	<p>Need to be provided as URL encoded post body. Only non-null values will be updated.</p> <ul style="list-style-type: none"> • <code><titleCode></code> (Required: false) • <code><firstName></code> (Required: false) • <code><lastName></code> (Required: false) • <code><language></code> (Required: false) • <code><currency></code> (Required: false)
Authentication	<ul style="list-style-type: none"> • Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first:

	<ul style="list-style-type: none"> ◦ <code>curl --location 'https://localhost:9002/authorizationserver/oauth' \</code> <code>--header 'Content-Type: application/x-www-form-urlencoded' \</code> <code>--data-urlencode 'username={uid}' \</code> <code>--data-urlencode 'password={pwd}' \</code> <code>--data-urlencode 'client_secret={clientSecret\}' \</code> <code>--data-urlencode 'grant_type=password' \</code> <code>--data-urlencode 'client_id={client_id}'</code> • HTTPS channel is required
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Representations	application/xml, application/json

Resource: /{site}/customers/current/addresses

Method	GET
Description	Get customer's addresses.
Example URL	https://localhost:9002/rest/v1/electronics/customers/current/addresses
Authentication	<ul style="list-style-type: none"> • Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> ◦ <code>curl --location 'https://localhost:9002/authorizationserver/oauth' \</code> <code>--header 'Content-Type: application/x-www-form-urlencoded' \</code> <code>--data-urlencode 'username={uid}' \</code> <code>--data-urlencode 'password={pwd}' \</code> <code>--data-urlencode 'client_secret={clientSecret\}' \</code> <code>--data-urlencode 'grant_type=password' \</code> <code>--data-urlencode 'client_id={client_id}'</code> • HTTPS channel is required
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Representations	application/xml, application/json

Resource: /{site}/customers/current/addresses

Method	POST
Description	Create new address.
Example URL	https://localhost:9002/rest/v1/electronics/customers/current/addresses
Request Parameters	<p>See <code>HttpRequestAddressDataPopulator</code> class. Need to be provided as URL encoded post body.</p> <ul style="list-style-type: none"> • <code><titleCode></code> (Required: true) • <code><firstName></code> (Required: true) • <code><lastName></code> (Required: true) • <code><line1></code> (Required: true) • <code><line2></code> (Required: false) • <code><town></code> (Required: true) • <code><postalCode></code> (Required: true)

	<ul style="list-style-type: none"> • <code><country.isocode></code> (Required: true) • <code><region.isocode></code> (Required: false)
Authentication	<ul style="list-style-type: none"> • Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> ◦ <code>curl --location 'https://localhost:9002/authorizationserver/oauth' \</code> <code>--header 'Content-Type: application/x-www-form-urlencoded' \</code> <code>--data-urlencode 'username={uid}' \</code> <code>--data-urlencode 'password={pwd}' \</code> <code>--data-urlencode 'client_secret={clientSecret\}' \</code> <code>--data-urlencode 'grant_type=password' \</code> <code>--data-urlencode 'client_id={client_id}'</code> • HTTPS channel is required
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: `/[site]/customers/current/addresses/{id}`

Method	PUT
Description	Update address.
Example URL	<code>https://localhost:9002/rest/v1/electronics/customers/current/addresses/12345</code>
Request Parameters	<p>See <code>HttpRequestAddressDataPopulator</code> class. Need to be provided as URL encoded post body. Only non-null values will be updated.</p> <ul style="list-style-type: none"> • <code><titleCode></code> (Required: false) • <code><firstName></code> (Required: false) • <code><lastName></code> (Required: false) • <code><line1></code> (Required: false) • <code><line2></code> (Required: false) • <code><town></code> (Required: false) • <code><postalCode></code> (Required: false) • <code><country.isocode></code> (Required: false) • <code><region.isocode></code> (Required: false)
Authentication	<ul style="list-style-type: none"> • Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> ◦ <code>curl --location 'https://localhost:9002/authorizationserver/oauth' \</code> <code>--header 'Content-Type: application/x-www-form-urlencoded' \</code> <code>--data-urlencode 'username={uid}' \</code> <code>--data-urlencode 'password={pwd}' \</code> <code>--data-urlencode 'client_secret={clientSecret\}' \</code> <code>--data-urlencode 'grant_type=password' \</code> <code>--data-urlencode 'client_id={client_id}'</code> • HTTPS channel is required
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.

	<ul style="list-style-type: none"> Content-Type application/x-www-form-urlencoded (Style: header, Required: true) application/x-www-form-urlencoded content type is required by HttpPutFormContentFilter
Representations	application/xml ,application/json

Resource: /{site}/customers/current/addresses/{id}

Method	DELETE
Description	Delete address.
Example URL	https://localhost:9002/rest/v1/electronics/customers/current/addresses/12345
Request Parameters	<ul style="list-style-type: none"> <id>(Style: body, Required: true): Address id.
Authentication	<ul style="list-style-type: none"> Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> curl --location 'https://localhost:9002/authorizationserver/oauth' \ --header 'Content-Type: application/x-www-form-urlencoded' \ --data-urlencode 'username={uid}' \ --data-urlencode 'password={pwd}' \ --data-urlencode 'client_secret={clientSecret\}' \ --data-urlencode 'grant_type=password' \ --data-urlencode 'client_id={client_id}' HTTPS channel is required
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml , application/json

Resource: /{site}/customers/current

Method	GET
Description	Returns customer profile.
Example URL	https://localhost:9002/rest/v1/electronics/customers/current
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Authentication	<ul style="list-style-type: none"> Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> curl --location 'https://localhost:9002/authorizationserver/oauth' \ --header 'Content-Type: application/x-www-form-urlencoded' \ --data-urlencode 'username={uid}' \ --data-urlencode 'password={pwd}' \ --data-urlencode 'client_secret={clientSecret\}' \ --data-urlencode 'grant_type=password' \ --data-urlencode 'client_id={client_id}' HTTPS channel is required

Representations	application/xml, application/json
-----------------	-----------------------------------

Resource: /{site}/customers/current/password

Method	POST
Description	Change customer's password.
Example URL	https://localhost:9002/rest/v1/electronics/customers/current/password
Request Parameters	<ul style="list-style-type: none"> • <old> (Style: body, Required: true): Old password. • <new> (Style: body, Required: true): New password.
Authentication	<ul style="list-style-type: none"> • Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> ◦ <code>curl --location 'https://localhost:9002/authorizationserver/oauth' \</code> <code>--header 'Content-Type: application/x-www-form-urlencoded' \</code> <code>--data-urlencode 'username={uid}' \</code> <code>--data-urlencode 'password={pwd}' \</code> <code>--data-urlencode 'client_secret={clientSecret\}' \</code> <code>--data-urlencode 'grant_type=password' \</code> <code>--data-urlencode 'client_id={client_id}'</code> • HTTPS channel is required
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml,application/json

Resource: /{site}/customers/current/forgottenpassword

Method	POST
Description	Restore customer's forgotten password.
Example URL	<ul style="list-style-type: none"> • Access is restricted to ROLE_CLIENT so that only registered API clients can gain it. The client access token must be sent with the request. To obtain the token for a given <client_id> an authorization call must be sent first: <ul style="list-style-type: none"> ◦ <code>curl --location 'https://localhost:9002/authorizationserver/oauth' \</code> <code>--header 'Content-Type: application/x-www-form-urlencoded' \</code> <code>--data-urlencode 'client_secret={clientSecret\}' \</code> <code>--data-urlencode 'grant_type=client_credentials' \</code> <code>--data-urlencode 'client_id={client_id}'</code> • Now you can send a call: <ul style="list-style-type: none"> ◦ <code>https://localhost:9002/rest/v1/electronics/customers/current/forgottenpassword</code>
Request Parameters	<ul style="list-style-type: none"> • <login> (Style: body, Required: true): Customer's login. Customer login is case insensitive.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true):

	Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: /{site}/customers/current/paymentinfos

Method	GET
Description	Return current customer's credit card payment infos that were previously saved during checkout.
Example URL	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Authentication	<ul style="list-style-type: none"> Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> curl --location 'https://localhost:9002/authorizationserver/oauth/token \ --header 'Content-Type: application/x-www-form-urlencoded' \ --data-urlencode 'username={uid}' \ --data-urlencode 'password={pwd}' \ --data-urlencode 'client_secret={clientSecret\}' \ --data-urlencode 'grant_type=password' \ --data-urlencode 'client_id={client_id}' HTTPS channel is required
Representations	application/xml, application/json

Resource: /{site}/customers/current/paymentinfos/{id}

Method	GET
Description	Return customer's credit card payment info by payment info id.
Example URL	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos/1234
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Authentication	<ul style="list-style-type: none"> Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> curl --location 'https://localhost:9002/authorizationserver/oauth/token \ --header 'Content-Type: application/x-www-form-urlencoded' \ --data-urlencode 'username={uid}' \ --data-urlencode 'password={pwd}' \ --data-urlencode 'client_secret={clientSecret\}' \ --data-urlencode 'grant_type=password' \ --data-urlencode 'client_id={client_id}' HTTPS channel is required
Representations	application/xml, application/json

Resource: /{site}/customers/current/paymentinfos/{id}

Method	DELETE
Description	Removes customer's credit card payment info by payment info ID.
Example URL	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos/1234
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Authentication	<ul style="list-style-type: none"> Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> curl --location 'https://localhost:9002/authorizationserver/oauth/token' \ --header 'Content-Type: application/x-www-form-urlencoded' \ --data-urlencode 'username={uid}' \ --data-urlencode 'password={pwd}' \ --data-urlencode 'client_secret={clientSecret\}' \ --data-urlencode 'grant_type=password' \ --data-urlencode 'client_id={client_id}' HTTPS channel is required
Representations	application/xml, application/json

Resource: /{site}/customers/current/paymentinfos/{id}

Method	PUT
Description	Updates existing customer's credit card payment info by payment info ID.
Example URL	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos/1234
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Request Parameters	<p>Need to be provided as URL encoded post body:</p> <ul style="list-style-type: none"> <accountHolderName> (Required: true): <cardNumber> (Required: true): <cardType> (Required: true): <p>Call GET /{site}/cardtypes beforehand to see what card types are supported.</p> <ul style="list-style-type: none"> <expiryMonth> (Required: true): <expiryYear> (Required: true): <saved> (Required: false): <p>Tells whether the payment info is saved for the customer and than could be reused for future orders.</p> <ul style="list-style-type: none"> <defaultPaymentInfo> (Required: false):
Authentication	<ul style="list-style-type: none"> Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> curl --location 'https://localhost:9002/authorizationserver/oauth/token' \ --header 'Content-Type: application/x-www-form-urlencoded' \ --data-urlencode 'username={uid}' \ --data-urlencode 'password={pwd}' \ --data-urlencode 'client_secret={clientSecret\}' \ --data-urlencode 'grant_type=password' \ --data-urlencode 'client_id={client_id}'

	<ul style="list-style-type: none"> • HTTPS channel is required
Representations	application/xml, application/json

Resource: /{site}/customers/current/paymentinfos/{id}/address

Method	POST
Description	Updates billing address of existing customer's credit card payment info by payment info ID.
Example URL	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos/1234/address
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Request Parameters	<p>Need to be provided as URL encoded post body:</p> <ul style="list-style-type: none"> • <titleCode> (Required: true): • <firstName> (Required: true): • <lastName> (Required: true): • <line1> (Required: true): • <line2> (Required: true): • <postalCode> (Required: true): • <town> (Required: true): • <country.isocode> (Required: true): <p>If the payment info should be saved for the customer and than could be reused for future orders.</p> <ul style="list-style-type: none"> • <defaultPaymentInfo> (Required: false):
Authentication	<ul style="list-style-type: none"> • Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> ◦ <pre>curl --location 'https://localhost:9002/authorizationserver/oauth/token \ --header 'Content-Type: application/x-www-form-urlencoded' \ --data-urlencode 'username={uid}' \ --data-urlencode 'password={pwd}' \ --data-urlencode 'client_secret={clientSecret\}' \ --data-urlencode 'grant_type=password' \ --data-urlencode 'client_id={client_id}'</pre> • HTTPS channel is required
Representations	application/xml, application/json

Resource: /{site}/customers/current/customergroups

Method	GET
Description	Returns all customer groups of current customer. Requires customergroup access role and a secured HTTPS channel.
Example URL	https://localhost:9002/rest/v1/electronics/customers/current/customergroups
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>

Authentication	<ul style="list-style-type: none"> Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> <code>curl --location 'https://localhost:9002/authorizationserver/oauth' \</code> <code>--header 'Content-Type: application/x-www-form-urlencoded' \</code> <code>--data-urlencode 'username={uid}' \</code> <code>--data-urlencode 'password={pwd}' \</code> <code>--data-urlencode 'client_secret={clientSecret\}' \</code> <code>--data-urlencode 'grant_type=password' \</code> <code>--data-urlencode 'client_id={client_id}'</code> HTTPS channel is required
Representations	application/xml, application/json

Resource: /{site}/customers/current/login

Method	POST
Description	Change customer's login.
Example URL	<code>https://localhost:9002/rest/v1/electronics/customers/current/login</code>
Request Parameters	<ul style="list-style-type: none"> <newLogin> (Style: body, Required: true): Customer's new login. Customer login is case insensitive. <password> (Style: body, Required: true): Customer's current password.
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: /{site}/customers/{uid}/customergroups

Method	GET
Description	Returns all customer groups of a given customer. Requires admingroup access role and a secured HTTPS channel.
Example URL	<code>https://localhost:9002/rest/v1/electronics/customers/demo/customergroups</code>
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Authentication	<ul style="list-style-type: none"> Access is restricted to ROLE_CUSTOMERGROUP so the customer access token must be sent with the request. To obtain the token for a given customer an authorization call must be sent first: <ul style="list-style-type: none"> <code>curl --location 'https://localhost:9002/authorizationserver/oauth' \</code> <code>--header 'Content-Type: application/x-www-form-urlencoded' \</code> <code>--data-urlencode 'username={uid}' \</code> <code>--data-urlencode 'password={pwd}' \</code> <code>--data-urlencode 'client_secret={clientSecret\}' \</code> <code>--data-urlencode 'grant_type=password' \</code> <code>--data-urlencode 'client_id={client_id}'</code> HTTPS channel is required

Representations	application/xml, application/json
-----------------	-----------------------------------

Resource Orders

A list of the available **Orders** resources.

i Note

The URL examples are used for presentation purposes only. In a customized environment, you must replace the server address with your own.

Resource: /{site}/orders

Method	GET
Description	Returns order history data for all orders placed by the current user for the current base store. Response contains orders search result displayed in several pages if needed.
Example URL	https://localhost:9002/rest/v1/electronics/orders
Authentication	This method requires basic HTTP authentication and is restricted to HTTPS channel.
Request Parameters	<p>As URL query parameters:</p> <ul style="list-style-type: none"> • <statuses> (Required: false) Filters only certain order statuses. It means: <statuses>=CANCELLED, CHECKED_VALID would only return orders with status CANCELLED or CHECKED_VALID. • <currentPage> (Required: false) Pagination attribute saying which page is requested • <pageSize> (Required: false) Pagination attribute saying the requested page size • <sort> (Required: false) Pagination attribute saying sort preference
Headers	<ul style="list-style-type: none"> • Accept(Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: /{site}/orders/{code}

Method	GET
Description	Returns specific order details. Response contains a detailed order info object.
Example URL	https://localhost:9002/rest/v1/electronics/orders/000001
Authentication	This method requires basic HTTP authentication and is restricted to HTTPS channel.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true):

	Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource Products

A list of the available **Products** resources.

i Note

The URL examples are used for presentation purposes only. In a customized environment, you must replace the server address with your own.

Resource: /{site}/products

Method	GET
Description	<p>Returns a list of products and additional data like available facets, available sort options, and pagination options. It can include spelling suggestions, for example:</p> <pre><spellingSuggestion> <suggestion>sony</suggestion> <query>sony:relevance</query> </spellingSuggestion></pre> <p>To make it work you need to:</p> <ul style="list-style-type: none"> Make sure <i><enableSpellCheck></i> on the <i>SearchQuery</i> is set to true. By default it should be already set to true. Have indexed properties configured to be used for spell checking.
Example URL	https://localhost:9002/rest/v1/electronics/products?query=a&pageSize=40
Authentication	None
Request Parameters	<ul style="list-style-type: none"> <i><query></i> (Style: query, Required: true): Serialized query, freetextsearch, facets. The format of serialized query: freeTextSearch:sort:facetKey1:facetValue1:facetKey2:facetValue2 <p>The query string needs to be URL-encoded and the client has to make sure that no : elements are part of the freetextsearch. The query value needs to be properly URL-encoded, too.</p> <ul style="list-style-type: none"> <i><currentPage></i> (Style: query, Required: false): The current result page requested. Optional. Default value: 0. <i><pageSize></i> (Style: query, Required: false): The number of results returned per page. Optional. Default value: 20. <i><sort></i> (Style: query, Required: false): Sorting method applied to the display search results.
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: `/[site]/products/[product_code]`

Method	GET
Description	Returns details of a single product specified by product code. Additional options can be expressed using options parameter.
Example URL	<code>https://localhost:9002/rest/v1/electronics/products/553637?options=DESCRIPTION</code>
Authentication	None
Request Parameters	<ul style="list-style-type: none"> • <i><options></i> (Style: query, Required: false): Defines level of the requested details. It can have a combination of the following values, separated by a comma: BASIC, DESCRIPTION, GALLERY, CATEGORIES, PROMOTIONS, STOCK, REVIEW, CLASSIFICATION, REFERENCES, PRICE. Keep in mind that the value needs to be properly URL-encoded. Example: <code>options=BASIC,REVIEW</code> requests the basic level of details plus reviews of a product.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be <code>application/xml</code> or <code>application/json</code>.
Representations	application/xml, application/json

Resource: `/[site]/products/export/incremental`

Method	GET
Description	Returns all products with pagination applied.
Example URL	<code>https://localhost:9002/rest/v1/electronics/products/export/incremental?timestamp=2012-03-28T07:50:49%2B00:00%tPage=0&pageSize=10&options=BASIC</code>
Authentication	<ul style="list-style-type: none"> • Access is restricted to the ROLE_TRUSTED_CLIENT so that only registered and trusted API clients can gain it. • HTTPS channel is required
Request Parameters	<ul style="list-style-type: none"> • <i><currentPage></i> (Style: query, Required: false): The current result page requested, optional. Default value: 0. • <i><timestamp></i> in RFC-8601 format • <i><pageSize></i> (Style: query, Required: false): The number of results returned per page. Optional. Default value: unlimited. • <i><options></i> (Style: query, Required: false): It defines the level of details requested. This parameter can have a combination of the following values, separated by a comma : BASIC, DESCRIPTION, GALLERY, CATEGORIES, PROMOTIONS, STOCK, REVIEW, CLASSIFICATION, REFERENCES. Keep in mind that the value needs to be properly URL-encoded. Example: <code>options=BASIC REVIEW</code> requests the basic level of details plus reviews of a product.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true):

	Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: /{site}/products/export/full

Method	GET
Description	Returns all products with pagination applied.
Example URL	https://localhost:9002/rest/v1/electronics/products/export/full?currentPage=0&pageSize=10&options=BASIC
Authentication	<ul style="list-style-type: none"> Access is restricted to the ROLE_TRUSTED_CLIENT so that only registered and trusted API clients can gain it. HTTPS channel is required
Request Parameters	<ul style="list-style-type: none"> <currentPage> (Style: query, Required: false): The current result page requested, optional. Default value: 0. <pageSize> (Style: query, Required: false): The number of results returned per page. Optional. Default value: unlimited. <options> (Style: query, Required: false): Defines the level of details requested. It can have a combination of the following values, separated by a comma : BASIC, DESCRIPTION, GALLERY, CATEGORIES, PROMOTIONS, STOCK, REVIEW, CLASSIFICATION, REFERENCES. Keep in mind that the value needs to be properly URL-encoded. Example: options=BASIC REVIEW requests the basic level of details plus reviews of a product.
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: /{site}/products/suggest

Method	GET
Description	<p>Related to spell check correction topic. It is a list of all available suggestions related to the given <term> and limits the results to a given value of the <max> parameter. It returns the result in the XML format.</p> <pre><suggestions> <suggestion> <value>tripod</value> </suggestion> <suggestion> <value>tape</value> </suggestion> </suggestions></pre>
Example URL	https://localhost:9002/rest/v1/electronics/products/suggest?term=t&max=3

Request Parameters	<ul style="list-style-type: none"> • <i><term></i> (Style: body, Required: true): Specified term • <i><max></i> (Style: body, Required: false): Specifies the limit of results. Default value: 10.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.

Resource: /{site}/products/{code}/reviews

Method	POST
Description	Method creates new customer review as an anonymous user. Method responses with a review data.
Example URL	https://localhost:9002/rest/v1/electronics/product/816780/reviews
Request Parameters	<p>Need to be provided as URL encoded post body.</p> <ul style="list-style-type: none"> • <i><rating></i> Required. Value needs to be between 1 and 5. • <i><alias></i> Optional • <i><headline></i> Required • <i><comment></i> Required
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: /{site}/products/expressUpdate

Method	GET
Description	Method returns products added to the express update feed. The queue is cleared using the defined cronjob.
Example URL	https://localhost:9002/rest/v1/electronics/products/expressUpdate?timestamp=2013-06-21T11%3A30%3A21%2B01%3A00
Authentication	<ul style="list-style-type: none"> • Access is restricted to the ROLE_TRUSTED_CLIENT so that only registered and trusted API clients can gain it. • HTTPS channel is required
Request Parameters	<ul style="list-style-type: none"> • <i><timestamp></i>: Required in RFC-8601 format. Only items newer than the given parameter are retrieved from the queue. • <i><catalog></i>: Optional. Only products from this catalog are returned. Format: catalogId:catalogVersion
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true):

	Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: /{site}/products/{code}/nearLocation

Method	GET
Description	Method returns product's stock level sorted by distance from specific location passed by the free-text parameter.
Example URL	<code>https://localhost:9002/rest/v1/electronics/products/816780/nearLocation?location=Tokio&pageSize=5&tPage=2</code>
Request Parameters	Need to be provided as URL encoded post body. <ul style="list-style-type: none"> • <i><location></i>: Required. Free-text location parameter. • <i><pageSize></i>: Optional • <i><currentPage></i>: Optional
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Representations	application/xml, application/json

Resource: /{site}/products/{code}/nearLatLong

Method	GET
Description	Method returns product's stock level sorted by distance from specific location passed by the longitude and latitude parameters.
Example URL	<code>https://localhost:9002/rest/v1/electronics/products/816780/nearLatLong?latitude=35.6816951&longitude=139.7650482&pageSize=5&tPage=2</code>
Request Parameters	Need to be provided as URL encoded post body. <ul style="list-style-type: none"> • <i><longitude></i>: Required. Longitude location parameter. • <i><latitude></i>: Required. Latitude location parameter. • <i><pageSize></i>: Optional • <i><currentPage></i>: Optional
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Representations	application/xml, application/json

Miscellaneous Resources

A list of available **Miscellaneous resources** that do not belong to other groups.

i Note

Resource: /{site}/languages

Method	GET
Description	List all available languages (all base store's languages). In case of an empty languages list for the base store, it returns list of all languages in the system.
Example URL	https://localhost:9002/rest/v1/electronics/languages
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>

Resource: /{site}/currencies

Method	GET
Description	List all available currencies (all usable currencies for the current store). In case of an empty currencies list for stores, it returns the list of all currencies in the system.
Example URL	https://localhost:9002/rest/v1/electronics/currencies
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>

Resource: /{site}/deliverycountries

Method	GET
Description	List all supported delivery countries for the current store. The list is sorted alphabetically.
Example URL	https://localhost:9002/rest/v1/electronics/deliverycountries
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>

Resource: /{site}/cardtypes

Method	GET
Description	List supported payment card types.
Example URL	https://localhost:9002/rest/v1/electronics/cardtypes

Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
---------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Resource: /{site}/titles

Method	GET
Description	List all localized titles.
Example URL	https://localhost:9002/rest/v1/electronics/titles
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>

Resource Stores

A list of available **Stores** resources.

i Note

The URL examples are used for presentation purposes only. In a customized environment, you must replace the server address with your own.

Resource: /{site}/stores

Method	GET
Description	<p>Lists all store locations that are near the location specified in a query or by latitude and longitude.</p> <pre><searchResult> <results /> <pagination> <pageSize>10</pageSize> <currentPage>0</currentPage> <totalResults>49</totalResults> <totalPages>5</totalPages> </pagination> <boundEastLongitude>135.2738964</boundEastLongitude> <boundSouthLatitude>33.5829778</boundSouthLatitude> <sourceLatitude>48.1366069</sourceLatitude> <boundNorthLatitude>62.69023599999999</boundNorthLatitude> <sourceLongitude>11.5770851</sourceLongitude> <locationText>munich</locationText> <boundWestLongitude>-112.1197262</boundWestLongitude> </searchResult></pre>
Example URL	https://localhost:9002/rest/v1/electronics/stores?query=munich https://localhost:9002/rest/v1/electronics/stores?query=munich
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>

Request Parameters	Not applicable.
Representations	application/xml, application/json

Resource: /{site}/stores/{name}

Method	GET
Description	Returns store location by its unique name.
Example URL	https://localhost:9002/rest/v1/electronics/stores/Chiba
Headers	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be application/xml or application/json.</p>
Request Parameters	Not applicable.
Representations	application/xml, application/json

Resource Promotions

A list of available **Promotions** resources.

i Note

The URL examples are used for presentation purposes only. In a customized environment, you must replace the server address with your own.

Resource: /{site}/promotions

Method	GET
Description	Method returns promotion list
Example URL	<ul style="list-style-type: none"> https://localhost:9002/rest/v1/electronics/promotions?type=all https://localhost:9002/rest/v1/electronics/promotions?type=all&promotionGroup=electronicsPromoGrp
Authentication	<ul style="list-style-type: none"> Access is restricted to the ROLE_CLIENT and ROLE_TRUSTED_CLIENT. Only registered API clients can gain it. The client access token must be sent with the request. To obtain the token for a given <i><client_id></i>, you need to send an authorization call first: <ul style="list-style-type: none"> POST - https://localhost:9002/rest/oauth/token <p>Header:</p> <p>Content-Type: application/x-www-form-urlencoded</p> <p>Request body:</p> <p>client_id=\${clientId}&client_secret=\${clientSecret}&grant_type=client_credentials</p> HTTPS channel is required
Request Parameters	<ul style="list-style-type: none"> <i><type></i> (Required: true): <p>Defines what type of promotions should be returned. Values supported for that parameter are:</p> <ul style="list-style-type: none"> <i><all></i>: All available promotions are returned <i><product></i>: Only product promotions are returned

	<ul style="list-style-type: none"> ◦ <code><order></code>: Only order promotions are returned • <code><promotionGroup></code> (Required: false): Only promotions from this group are returned
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource: `/[site]/promotions/{code}`

Method	GET
Description	Returns details of a single promotion specified by a promotion code. Additional options can be expressed by using the <code><options></code> parameter.
Example URL	<code>https://localhost:9002/rest/v1/electronics/promotions/MultiBuy?options=EXTENDED</code>
Authentication	Access is restricted to the ROLE_CLIENT and ROLE_TRUSTED_CLIENT . This method requires client HTTP authentication and is restricted to HTTPS channel.
Request Parameters	<ul style="list-style-type: none"> • <code><options></code> (Style: query, Required: false): Defines level of the requested details. It can have a combination of the following values, separated by comma: BASIC, EXTENDED. The <code><BASIC></code> value is included by default. The value needs to be properly URL-encoded. Example: <code>options=BASIC, EXTENDED</code> requests the basic level of details plus extended.
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): Accept header needs to be sent with each request. It can be application/xml or application/json.
Representations	application/xml, application/json

Resource Vouchers

A list of available **Vouchers** resources.

i Note

The URL examples are used for presentation purposes only. In a customized environment, you must replace the server address with your own.

Resource: `/[site]/vouchers/{code}`

Method	GET
Description	Returns details of a single voucher specified by a voucher code.
Example URL	<code>https://localhost:9002/rest/v1/electronics/vouchers/abc-9PSW-EDH2-RXKA</code>
Authentication	<ul style="list-style-type: none"> • Access is restricted to the ROLE_CLIENT so that only registered API clients can gain it. The client access token must be sent with the request. To obtain the token for a given <code><client_id></code>, you need to send an authorization call

	<p>first: POST - https://localhost:9002/authorizationserver/oauth/token</p> <p>Header:</p> <p>Content-Type: <code>application/x-www-form-urlencoded</code></p> <p>Request body:</p> <p><code>client_id=\${clientId}&client_secret=\${clientSecret}&grant_type=client_credentials</code></p> <ul style="list-style-type: none"> • HTTPS channel is required
Headers	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): <p>Accept header needs to be sent with each request. It can be <code>application/xml</code> or <code>application/json</code>.</p>
Representations	<code>application/xml</code>, <code>application/json</code>

Resource WCMS Components

A list of the available **WCMS Components** resources.

i Note

The URL examples are used for presentation purposes only. In a customized environment, you must replace the server address with your own.

Resource: `/[site]/wcms/images/[code]`

Method	GET
Validity	5.0.1
Description	Returns details of a single CMSImageComponent specified by code.
Example URL	https://localhost:9002/rest/v1/electronics/wcms/images/image_code
Request Parameters	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): <p>It can be <code>application/xml,application/json, image/*</code></p> <p>If an Accept header is set to <code>image/*</code> and component is not visible, then the server returns the 404 (Not Found) error.</p> <p>If an Accept header field is present, and if the server cannot send a response which is acceptable according to the combinedAccept field value, then the server returns the 406 (Not Acceptable) error.</p> <p>If no Accept header field is present, then the JSON representation is returned.</p>
Representations	<ul style="list-style-type: none"> • Accept (Style: header, Required: true): <p>It can be <code>application/xml, application/json, image/*</code></p>

Resource: `/[site]/wcms/paragraphs/[code]`

Method	GET
Validity	5.0.1
Description	Returns details of a single CMSParagraphComponent specified by code.
Example URL	https://localhost:9002/rest/v1/electronics/wcms/paragraphs/paragraph_code
Request Parameters	<ul style="list-style-type: none"> • Accept

	<p>header needs to be sent with each request. It can be application/xml, application/json, text/html</p> <p>If an Accept header is set to text/html and component is not visible, then the server returns the 404 (Not Found) error.</p> <p>If an Accept header field is present, and if the server cannot send a response which is acceptable according to the combined Accept</p> <p>field value, then the server returns the 406 (Not Acceptable) error.</p> <p>If no Accept header field is present, then the JSON representation is returned.</p>
Representations	<ul style="list-style-type: none"> Accept (Style: header, Required: true): <p>It can be application/xml, application/json, text/html</p>

OCC v1 Sample Flows

Sample flow calls for the SAP Commerce OCC API in **v1**

Overview

i Note

The URL examples are used for presentation purposes only. In a customized environment, you must replace the server address with your own.

In general, REST requests are stateless, but the `ycommercewebservices` extension uses a standard HTTP session to simplify calls when operating on a customer cart. It is the same approach as the end-user session on a web front end.

Despite that a session is needed only for a few calls, it is recommended to always use it. The server creates a new session and sends back a session identifier in response to the first request. It is good practice to store a session ID and use it in the following calls every time the server generates it. Using a session ID from the beginning also simplifies the application because there is no need to analyze if it should be remembered or not.

HTTP Session

A session is a sequence of operations (request, response) done by one user. Usually these operations share and use some common data, for example a customer cart or information about a logged-in user. The session is started during the first request and is finished when the user logs out or after a long time of inactivity.

HTTP protocol is not session-aware and particular requests are linked to its session based on the session identifier which is sent as a cookie with each request.

Every time a server sends a new session identifier (JSESSIONID cookie in java applications), the client application should remember it and use it in the following calls, otherwise a new session is created each time.

Example:

End-user would like to insert a product to a cart. The client application sends a request:

Request #1

POST `https://localhost:9002/rest/v1/electronics/cart`

POST data:
code=3429337

Cookie data:

This is the first request, so the client application does not have any cookies. The server does not receive a JSESSIONID cookie and assumes that this is a new session. It creates the session and sends its identifier to the client application in **Set-Cookie** header:

Set-Cookie: JSESSIONID=911CFAE8289BFB29AB347EC95B874EF7; Path=/rest/; HttpOnly

The client application should remember this cookie and resend it with subsequent requests to this server. So for example, if an end-user would like to get his cart contents, the application sends a request:

Request #2

GET https://localhost:9002/rest/v1/electronics/cart

Cookie Data:

JSESSIONID=911CFAE8289BFB29AB347EC95B874EF7

The server receives the JSESSIONID and finds the appropriate session that contains the cart contents of the current user and returns it as the response.

Cookie Attributes

Cookies are defined as a key-value pair with some additional attributes:

Set-Cookie: JSESSIONID=911CFAE8289BFB29AB347EC95B874EF7; Path=/rest/; Secure; HttpOnly

- **<Path>** attribute sets the cookie 'visibility range'. So in this case, the JSESSIONID cookie will be used only with requests to URLs starting with **/rest**.
- **<Secure>** flag indicates that this cookie may be transmitted only with a secure communication channel.

Session ID Change

Session identifier may change when a new security level changes, in particular when:

- The user logs in

This is protection against **Session fixation** attack.

- The protocol changes from HTTPS to HTTP

This is protection against the **Man-in-the-middle** attack. When a cookie is generated while the HTTPS protocol is used, the server sets a **Secure** flag in this cookie. It then cannot be used with the HTTP protocol as described before, so a new JSESSIONID is generated in this case. But in general, when an application switches to an HTTPS protocol, it should not switch back to HTTP until the session ends (user logout, etc.). **The simplest rule is to always use HTTPS channel.**

HTTP Persistent Connection

For performance reasons, it is important to reuse existing connections, especially when subsequent communication is expected. It is very important when using the HTTPS protocol because creating a secure connection needs more time and more network communication.

When using an HTTP 1.0 client that supports a persistent connection, send the following header with the request:

Connection: Keep-Alive

If the server puts the same header in the response, the client may use this connection with the following calls.

In HTTP 1.1, connections are considered persistent by default unless declared otherwise.

Despite the connection being 'persistent', it does not mean that it is kept alive forever. It should be closed by the client or server after a period of inactivity.

Multiple Site Issues

It is possible to have multiple sites supported by one server instance. In every REST call, the particular site is specified in the URL. For example:

GET https://localhost:9002/rest/v1/electronics/cart

GET https://localhost:9002/rest/v1/apparel-uk/cart

It is important that some entities are common while others are separated depending on the site.

User accounts are common, so it is not possible to have two user accounts with the same login linked to other sites. Also, a user created in one site will be able to use the same account in other sites.

The OAuth access token is also connected to the user and is not dependent on the site. This is why access tokens are obtained using a URL which does not contain the site ID:

GET https://localhost:9002/authorizationserver/oauth/token

Once obtained, add the access token to the header, for example:

```
header 'Authorization: Bearer abc738b9-b930-4956-b2f0-f81665f78785'
```

The access token may then be used with REST calls for various sites:

GET https://localhost:9002/rest/v1/electronics/customers/current

GET https://localhost:9002/rest/v1/apparel-uk/customers/current

Orders

Order history is separated depending on the store. For example, when a user calls:

GET https://localhost:9002/rest/v1/electronics/orders?access_token=abc738b9-b930-4956-b2f0-f81665f78785

It returns orders placed in the electronics store (connected with the requested electronics site).

Carts

A user's cart is connected with a current user's session. In order to make it work properly across multiple sites it is very important to use separate HTTP sessions for each site.

To learn more about single steps for buying scenarios see: [Single Steps of Buying Process in v1](#).

To go through complete scenarios visit: [Customer Buying Process Scenarios in v1](#)

Single Steps of Buying Process in v1

Examples of single steps of Buying Process Scenarios for the Omni Commerce Connect (OCC) in v1.

Overview

The sections below present sets of available calls that may be executed using the v1 API. The calls have been grouped into topics reflecting their purpose, however they should be treated as a separate examples rather than complete scenarios. For complete business scenarios, refer to: [Customer Buying Process Scenarios in v1](#).

Register New Customer

Method	URL	Body(url encoded)
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ID&client_secret=\$CLIENT_SECRET&grant_type=client_credentials
POST	https://localhost:9002/rest/v1/electronics/customers	login=john.doe@mail.com&password=your_password &firstName=John&lastName=Doe&titleCode=mr

Get Token for Customer

Method	URL	Body(url encoded)
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ID\$&client_secret=\$CLIENT_SECRET\$ &grant_type=password&username=john.doe@mail.com&password=your_

Manage Customer Profile

Scenario steps:

- Get token for customer
- Manage addresses
- Manage payment information
- Update customer profile
- Change customer password

Method	URL	Body(url encoded)
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ &grant_type=passw
GET	https://localhost:9002/rest/v1/electronics/customers/current/addresses	
POST	https://localhost:9002/rest/v1/electronics/customers/current/addresses	firstName=John&las 19&town=Osaka&pc 27
PUT	https://localhost:9002/rest/v1/electronics/customers/current/addresses/default/8796158590999	
GET	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos	saved=true
GET	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos/8796191359018	
DELETE	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos/8796191359018	
PUT	https://localhost:9002/rest/v1/electronics/customers/current/profile	titleCode=dr&firstN:

Method	URL	Body(url encoded)
PUT	https://localhost:9002/rest/v1/electronics/customers/current/password	old=your_password&new=your_new_password

Search for a Product

Method	URL	Parameters	Descr
GET	https://localhost:9002/rest/v1/electronics/products	query=:price-asc:category:575	Return products from category with ID 575 sorted by ascending price.
GET	https://localhost:9002/rest/v1/electronics/products	query=:name-desc:brand:brand_10	Return products from category with ID 575 sorted by descending brand name.
GET	https://localhost:9002/rest/v1/electronics/products	query=camera&pageSize=40	Search for 'camera' products. We can narrow the response to 40 results.
GET	https://localhost:9002/rest/v1/electronics/products	query= camera:relevance:category:575	Search for 'camera' products in category with ID 575.
GET	https://localhost:9002/rest/v1/electronics/products	query= camera:relevance:category:575:brand:brand_10	Search for 'camera' products from category with ID 575 and brand ID of brand_10.
GET	https://localhost:9002/rest/v1/electronics/products	query= camera:relevance:category:575:brand:brand_10:price:\$500-\$999.99	Search for 'camera' products in category with ID 575 and brand ID of brand_10 and with price between \$500 and \$999.99.

Method	URL	Parameters	Descr
			betwe 500 a 999.9'

Search for Nearest Store Having Particular Product in Stock

Scenario steps:

- Search for product
- Search for nearest store having this product in stock

Method	URL	Parameters	Description
GET	https://localhost:9002/rest/v1/electronics/products	query=EOS	Searches for products containing the 'EOS' string in the name.
GET	https://localhost:9002/rest/v1/electronics/products/1382080		Gets details about 'Canon EOS 450D'.
GET	https://localhost:9002/rest/v1/electronics/products/1382080/nearLocation	location=Tokio	Searches for stores having 'Canon EOS 450D' in stock. The result is sorted by distance from given location.

Adding Product to Cart

Caution

Remember to set **JSESSIONID** in cookies. It is required for requests that change the cart. You will get it in Set-Cookie header in response.

Method	URL	Parameters	Description
GET	https://localhost:9002/rest/v1/electronics/cart		Gets an empty cart.
POST	https://localhost:9002/rest/v1/electronics/cart/entry	code=3429337	Adds product with code 3429337 to the cart.
POST	https://localhost:9002/rest/v1/electronics/cart/entry	code=1934795&qty=2	Adds two products with code 1934795 to the cart.

Method	URL	Parameters	Description
POST	https://localhost:9002/rest/v1/electronics/cart/entry	code=1934795&qty=1&storeName=Nakano	Adds one product with code 1934795 to the cart. Product will be picked up in Nakano store.
PUT	https://localhost:9002/rest/v1/electronics/cart/entry/0	qty=3	Changes quantity for first product in the cart from 1 to 3.
PUT	https://localhost:9002/rest/v1/electronics/cart/entry/2/store	storeName=Tokio	Changes store where product will be picked up.
DELETE	https://localhost:9002/rest/v1/electronics/cart/entry/2/store		Resets the store where the entry should be picked up. Entry is delivered by the selected delivery method.
DELETE	https://localhost:9002/rest/v1/electronics/cart/entry/0		Removes first product from the cart.
GET	https://localhost:9002/rest/v1/electronics/cart		Gets an updated cart.

Checkout Process

Checkout with Creating New Address and Payment Information for Customer

Assumption: a customer is registered and has a cart with products for current session.

Caution

Remember to set **JSESSIONID** in cookies. It is required for requests that make changes to the cart. You will get it in the Set-Cookie header in the response.

Scenario steps:

- Create address and set it as delivery address
- Set delivery mode
- Create payment information
- Authorize cart
- Place order

Method	URL	Request Body
POST	https://localhost:9002/rest/v1/electronics/customers/current/addresses	
PUT	https://localhost:9002/rest/v1/electronics/cart/address/delivery/8796158787607	
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/premium-gross	
POST	https://localhost:9002/rest/v1/electronics/cart/authorize	
POST	https://localhost:9002/rest/v1/electronics/cart/placeorder	
POST	https://localhost:9002/rest/v1/electronics/cart/paymentinfo	<p>Should set header Content-Type to : application/x-www-form-urlencoded</p> <pre> accountHolderName: "John Doe" cardNumber: "4111111111111111" cardType: "visa" expiryMonth: "01" expiryYear: "2017" saved: "true" defaultPaymentInfo: "true" billingAddress.titleCode: "mr" billingAddress.firstName: "John" billingAddress.lastName: "Doe" billingAddress.line1: "Toyosaki" billingAddress.line2: "3-16-19" billingAddress.postalCode: "555" billingAddress.town: "Osaka" billingAddress.country.isocode: "JP" billingAddress.region.isocode: "JP-26" </pre>

Checkout with Setting Existing Address and Payment Information for Customer

Assumption: a customer is registered and has a cart with products for current session.

Caution

Remember to set **JSESSIONID** in cookies. It is required for requests that make changes to the cart. You will get it in the Set-Cookie header in the response.

Scenario steps :

- Try to get delivery modes
- Set delivery address
- Set delivery mode
- Set payment information
- Authorize cart

- Place order

Method	URL	Parameters	Authorization	Description
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes		OAUTH token for customer	Get delivery modes for the current cart
GET	https://localhost:9002/rest/v1/electronics/customers/current/addresses		OAUTH token for customer	Get current customer addresses
PUT	https://localhost:9002/rest/v1/electronics/cart/address/delivery/8796158754839		OAUTH token for customer	Set delivery address for the current cart
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes		OAUTH token for customer	Get delivery modes for the current cart
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/premium-gross		OAUTH token for customer	Set premium gross delivery mode for the current cart
GET	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos	saved=true	OAUTH token for customer	Get current customer payment information
PUT	https://localhost:9002/rest/v1/electronics/cart/paymentinfo/8796158591018		OAUTH token for customer	Set payment information for the current cart
POST	https://localhost:9002/rest/v1/electronics/cart/authorize	securityCode=123	OAUTH token for customer	Authorize the current cart
POST	https://localhost:9002/rest/v1/electronics/cart/placeorder		OAUTH token for customer	Place order for the current cart

Get Customer Orders

Method	URL	Body(url encoded)
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ID\$&client_secret=\$CLIENT_SECRET\$ &grant_type=password&username=john.doe@mail.com&password=
GET	https://localhost:9002/rest/v1/electronics/orders	

Method	URL	Body(url encoded)
GET	https://localhost:9002/rest/v1/electronics/orders/00001002	

Store Search, Display Store Details

Scenario steps:

- Search for store
- Display store details

Method	URL	Parameters	Description
GET	https://localhost:9002/rest/v1/electronics/stores	query=Tokio	Searches for stores that are near the location specified by the parameters (in this case near Tokio city). Returns list of stores.
GET	https://localhost:9002/rest/v1/electronics/stores/Nakano		Gets store details for a store identified by name (in this case "Nakano").

Cart Restoration

Scenario steps:

- Search for products, session "1" is created automatically
- Get product details
- Insert product(s) to cart, cart "1" is created automatically
- Get cart in order to show it and remember cart **guid**
- Session 1 expires after long inactivity
- Get cart in order to show it, session "2" and cart "2" are created automatically
- Try to restore cart "1" to the current session "2"
- Get cart "1"

Method	URL	Parameters	Cookies
GET	https://localhost:9002/rest/v1/electronics/products	query=camera&pageSize=40	

Method	URL	Parameters	Cookies
GET	https://localhost:9002/rest/v1/electronics/products/489702		JSESSIONID obtained in step 1
POST	https://localhost:9002/rest/v1/electronics/cart/entry	code=489702	JSESSIONID obtained in step 1
GET	https://localhost:9002/rest/v1/electronics/cart		JSESSIONID obtained in step 1
-	-	-	-
GET	https://localhost:9002/rest/v1/electronics/cart		JSESSIONID obtained in step 1
GET	https://localhost:9002/rest/v1/electronics/cart/restore	guid=02017b6dad8834ba6f3fd26eb9ab4f270e9b1858	JSESSIONID obtained in step 6

Method	URL	Parameters	Cookies
GET	https://localhost:9002/rest/v1/electronics/cart		JSESSIONID obtained in step 7

Customer Buying Process Scenarios in v1

The following provides full Customer Buying Process Scenarios for Omni Commerce Connect (OCC) in v1.

Overview

The scenarios provided below represent a complete step-by-step guidance on using a given sequence of calls.

Registered Customer Scenario

Prerequisites: a customer is already registered in the store and has defined all information needed for the checkout process (such as address and payment information).

Caution

Remember to set **JSESSIONID** in cookies. It is required for requests that make changes to the cart. You get it in the Set-Cookie header in the response.

Scenario steps:

- Search product
- Get product details
- Add product to cart (basket)
- Get access token for customer
- Checkout
- Display placed order for customer
- Log out

Note

The calls should be executed in the order defined in the request flow table. Scroll to the right to see the full table content.

Method	URL	Body (url encoded)
GET	https://localhost:9002/rest/v1/electronics/products	query=camera&pageSize=40
GET	https://localhost:9002/rest/v1/electronics/products /489702	
POST	https://localhost:9002/rest/v1/electronics/cart/entry	code=489702

Method	URL	Body (url encoded)
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ID&client_secret=\$ &grant_type=password&username=johr
GET	https://localhost:9002/rest/v1/electronics/cart	
GET	https://localhost:9002/rest/v1/electronics/customers/current/addresses	
PUT	https://localhost:9002/rest/v1/electronics/cart/address/delivery/8796158754839	
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/premium-gross	
GET	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos	saved=true
PUT	https://localhost:9002/rest/v1/electronics/cart/paymentinfo/8796158591018	
POST	https://localhost:9002/rest/v1/electronics/cart/authorize	securityCode=123
POST	https://localhost:9002/rest/v1/electronics/cart/placeorder	
GET	https://localhost:9002/rest/v1/electronics/orders	
GET	https://localhost:9002/rest/v1/electronics/orders/00001002	
POST	https://localhost:9002/rest/v1/customers/current/logout	

New Customer Scenario

Prerequisites: a customer is not registered in the store.

This is custom documentation. For more information, please visit the [SAP Help Portal](#).

⚠ Caution

Remember to set **JSESSIONID** in cookies. It is required for requests that make changes to the cart. You will get it in the Set-Cookie header in the response.

Scenario steps:

- Register a new customer and get access token
- Create a new address for the customer
- Search for a specific product
- Get product details
- Add products to the cart
- Checkout: set delivery address, set delivery mode, set credit card information (paymentinfo), card authorization
- Place order
- Display placed order for the customer
- Log out

i Note

The calls should be executed in the order defined in the request flow table. Scroll to the right to see the full table content.

Method	URL	Request Body
POST	https://localhost:9002/authorizationserver/oauth/token	
POST	https://localhost:9002/rest/v1/electronics/customers	
POST	https://localhost:9002/authorizationserver/oauth/token	
POST	https://localhost:9002/rest/v1/electronics/customers/current/addresses	
GET	https://localhost:9002/rest/v1/electronics/products	
GET	https://localhost:9002/rest/v1/electronics/products /3429337	
POST	https://localhost:9002/rest/v1/electronics/cart/entry	

Method	URL	Request Body
PUT	https://localhost:9002/rest/v1/electronics/cart/address/delivery/8796158754839	
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/premium-gross	
POST	https://localhost:9002/rest/v1/electronics/cart/authorize	
POST	https://localhost:9002/rest/v1/electronics/cart/placeorder	
GET	https://localhost:9002/rest/v1/electronics/orders	
GET	https://localhost:9002/rest/v1/electronics/orders/00001002	
POST	https://localhost:9002/rest/v1/customers/current/logout	
POST	https://localhost:9002/rest/v1/electronics/cart/paymentinfo	<p>Should set Head Content-Type to: application/x-www-form-urlencoded</p> <pre> accountHolderName: "John Doe" cardNumber: "4111111111111111" cardType: "visa" expiryMonth: "01" expiryYear: "2017" saved: "true" defaultPaymentInfo: "true" billingAddress.titleCode: "mr" billingAddress.firstName: "John" billingAddress.lastName: "Doe" billingAddress.line1: "Toyosaki" billingAddress.line2: "3-16-15" billingAddress.postalCode: "530" billingAddress.town: "Osaka" billingAddress.country.isocode: "JP" billingAddress.region.isocode: "JP-10" </pre>

Pick Up in Store Scenario

Prerequisites: Customer is already registered in the store and has defined all information needed for the checkout process (like payment information).

Caution

Remember to set **JSESSIONID** in cookies. It is required for requests that make changes to the cart. You will get it in the Set-Cookie header in the response.

Scenario steps:

- Get an access token for the customer
- Add products that will be picked up in the store
- Checkout and set 'pickup' as the delivery mode
- Display placed order for the customer
- Log out

i Note

The calls should be executed in the order defined in the request flow table. Scroll to the right to see the full table content.

Method	URL	Body Parameters
		Content-Type: application/x-www-form-urlencoded
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ID&client_secret=\$CLIEN
POST	https://localhost:9002/rest/v1/electronics/cart/entry	code=1382080
PUT	https://localhost:9002/rest/v1/electronics/cart/entry/0/store	storeName=Nakano
POST	https://localhost:9002/rest/v1/electronics/cart/entry	code=1934795&qty=1&storeName=Nakano
GET	https://localhost:9002/rest/v1/electronics/cart	
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/pickup	
GET	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos	saved=true
PUT	https://localhost:9002/rest/v1/electronics/cart/paymentinfo/8796158591018	
POST	https://localhost:9002/rest/v1/electronics/cart/authorize	securityCode=123
POST	https://localhost:9002/rest/v1/electronics/cart/placeorder	

Method	URL	Body Parameters
		Content-Type: application/x-www-form-urlencoded
GET	https://localhost:9002/rest/v1/electronics/orders	
GET	https://localhost:9002/rest/v1/electronics/orders/00002036	
POST	https://localhost:9002/rest/v1/customers/current/logout	

Pick Up in Store with Store Consolidation - Scenario

Prerequisites:

- Customer is already registered in the store and has defined all information needed for the checkout process (such as address and payment information)
- acceleratorwebservicesaddon** is installed

Caution

Remember to set **JSESSIONID** in cookies. It is required for requests that make changes to the cart. You will get it in the Set-Cookie header in the response.

Scenario steps:

- Get an access token for customer
- Add a product that will be picked up in the store
- Add a product that will be picked up in a different store
- Use consolidate store functionality
- Checkout
- Log out

Note

The calls should be executed in the order defined in the request flow table. Scroll to the right to see the full table content.

Method	URL	Body Parameters
		Content-Type: application/x-www-form-urlencoded
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ID&client_secret=\$CLIENT_SECRET
POST	https://localhost:9002/rest/v1/electronics/cart/entry	code=2053226&qty=1&storeName=Shinbashi
POST	https://localhost:9002/rest/v1/electronics/cart/entry	code=1934793&qty=1&storeName=Nakano
GET	https://localhost:9002/rest/v1/electronics/cart	

Method	URL	Body Parameters
		Content-Type: application/x-www-form-urlencoded
GET	https://localhost:9002/rest/v1/electronics/cart/consolidate	
POST	https://localhost:9002/rest/v1/electronics/cart/consolidate	storeName=Shinbashi
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/pickup	
GET	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos	saved=true
PUT	https://localhost:9002/rest/v1/electronics/cart/paymentinfo/8796158591018	
POST	https://localhost:9002/rest/v1/electronics/cart/authorize	securityCode=123
POST	https://localhost:9002/rest/v1/electronics/cart/placeorder	
POST	https://localhost:9002/rest/v1/customers/current/logout	

Combining Delivery Mode With Pick Up in Store - Scenario

Prerequisites: Customer is already registered in the store and has defined all information needed for checkout process (such as address and payment information).

Caution

Remember to set **JSESSIONID** in cookies. It is required for requests that make changes in the cart. You will get it in the Set-Cookie header in the response.

Scenario steps:

- Get an access token for the customer
- Add a product that will be delivered
- Add a product that will be picked up in a store
- Checkout

- Display placed order for the customer
- Log out

i Note

The calls should be executed in the order defined in the request flow table. Scroll to the right to see the full table content.

Method	URL	Body Parameters
		Content-Type: application/x-www-form
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ID&client_secret=\$
POST	https://localhost:9002/rest/v1/electronics/cart/entry	code=1382080
POST	https://localhost:9002/rest/v1/electronics/cart/entry	code=1934795&qty=1&storeName=Nak
GET	https://localhost:9002/rest/v1/electronics/cart	
GET	https://localhost:9002/rest/v1/electronics/customers/current/addresses	
PUT	https://localhost:9002/rest/v1/electronics/cart/address/delivery/8796192014359	
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/premium-gross	
GET	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos	saved=true
PUT	https://localhost:9002/rest/v1/electronics/cart/paymentinfo/8796158591018	
POST	https://localhost:9002/rest/v1/electronics/cart/authorize	securityCode=123
POST	https://localhost:9002/rest/v1/electronics/cart/placeorder	
GET	https://localhost:9002/rest/v1/electronics/orders	

Method	URL	Body Parameters
		Content-Type: application/x-www-form
GET	https://localhost:9002/rest/v1/electronics/orders/00002036	
POST	https://localhost:9002/rest/v1/customers/current/logout	

Guest Checkout Scenario with Creating Full Account

⚠ Caution

Remember to set **JSESSIONID** in cookies. It is required for requests that make changes to the cart. You will get it in the Set-Cookie header in the response.

Scenario steps:

- Search a product
- Get the product details
- Add product to cart
- Get a client access token
- Introduce oneself as a guest
- Checkout
- Display placed order
- Convert guest account to full customer account (optional)
- Log in as a customer
- Get order list

i Note

The calls should be executed in the order defined in the request flow table. Scroll to the right to see the full table content.

Method	URL	Body Parameters
		Content-Type: application/x-www-form
GET	https://localhost:9002/rest/v1/electronics/products	
GET	https://localhost:9002/rest/v1/electronics/products /489702	
POST	https://localhost:9002/rest/v1/electronics/cart/entry	
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ID&client_secret=\$
POST	https://localhost:9002/rest/v1/electronics/customers/current/guestlogin	

Method	URL	Body Parameters
		Content-Type: application/x-www-form
POST	https://localhost:9002/rest/v1/electronics/customers/current/addresses	
PUT	https://localhost:9002/rest/v1/electronics/cart/address/delivery/8796158754839	
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/premium-gross	
POST	https://localhost:9002/rest/v1/electronics/cart/authorize	securityCode=123
POST	https://localhost:9002/rest/v1/electronics/cart/placeorder	
POST	https://localhost:9002/rest/v1/electronics/customers/current/convert	password=your_password
POST	https://localhost:9002/rest/v1/customers/current/logout	
	...	
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ID&client_secret=\$ &grant_type=password&username=johr
GET	https://localhost:9002/rest/v1/electronics/orders	
GET	https://localhost:9002/rest/v1/electronics/orders/00002009	
POST	https://localhost:9002/rest/v1/electronics/cart/paymentinfo	accountHolderName: "John Doe" cardNumber: "4111111111111111" cardType: "visa" expiryMonth: "01" expiryYear: "2017" saved: "true" defaultPaymentInfo: "true" billingAddress.titleCode: "mr" billingAddress.firstName: "Jo billingAddress.lastName: "Doe" billingAddress.line1: "Toyosa

Method	URL	Body Parameters
		Content-Type: application/x-www-form
		billingAddress.line2: "3-16-19 billingAddress.postalCode: "55 billingAddress.town: "Osaka" billingAddress.country.isocode billingAddress.region.isocode:

Guest Checkout Scenario with Checking Order Status

⚠ Caution

Remember to set **JSESSIONID** in cookies. It is required for requests that make changes to the cart. You will get it in the Set-Cookie header in the response.

Scenario steps:

- Search a product
- Get the product details
- Add a product to cart
- Get a client access token
- Introduce oneself as a guest
- Checkout
- Log out
- Get order information

i Note

The calls should be executed in the order defined in the request flow table. Scroll to the right to see the full table content.

Method	URL	Body Parameters
		Content-Type
GET	https://localhost:9002/rest/v1/electronics/products	
GET	https://localhost:9002/rest/v1/electronics/products /489702	
POST	https://localhost:9002/rest/v1/electronics/cart/entry	
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$Cl
POST	https://localhost:9002/rest/v1/electronics/customers/current/guestlogin	
POST	https://localhost:9002/rest/v1/electronics/customers/current/addresses	

Method	URL	Body Parame
		Content-Type
PUT	https://localhost:9002/rest/v1/electronics/cart/address/delivery/8796158754839	
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/premium-gross	
POST	https://localhost:9002/rest/v1/electronics/cart/authorize	securityCode:
POST	https://localhost:9002/rest/v1/electronics/cart/placeorder	
POST	https://localhost:9002/rest/v1/customers/current/logout	
	...	
GET	https://localhost:9002/rest/v1/electronics/orders/byGuid/00aa0ba88c9c7c93b886423e354556dc21c430e0	
POST	https://localhost:9002/rest/v1/electronics/cart/paymentinfo	accountHo cardNumbe cardType: expiryMon expiryYea saved: "t defaultPa billingAd billingAd billingAd billingAd billingAd billingAd billingAd billingAd billingAd billingAd billingAd

Voucher Usage Scenario

Prerequisites: A voucher with code HY2008 is defined in the system.

Caution

Remember to set **JSESSIONID** in cookies. It is required for requests that make changes to the cart. You will get it in the Set-Cookie header in the response.

Scenario steps:

- Add products to the cart
- Apply the voucher
- Get cart
- Register the new customer and get an access token for it
- Create a new address for the customer
- Checkout: set delivery address, set delivery mode, set credit card information (payment info), card authorization
- Place order
- Display placed orders for the customer
- Log out

i Note

The calls should be executed in the order defined in the request flow table. Scroll to the right to see the full table content.

Method	URL	Body Parameters
		Content-Type: application/x-www-
POST	https://localhost:9002/rest/v1/electronics/cart/entry	
GET	https://localhost:9002/rest/v1/electronics/cart	
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ID\$&client_secret=\$
POST	https://localhost:9002/rest/v1/electronics/cart/voucher/HY2008	
GET	https://localhost:9002/rest/v1/electronics/cart	
POST	https://localhost:9002/rest/v1/electronics/customers	login=john.doe%40mail.com&password &firstName=John&lastName=Doe&title=
POST	https://localhost:9002/authorizationserver/oauth/token	client_id=\$CLIENT_ID\$&client_secret=\$
POST	https://localhost:9002/rest/v1/electronics/customers/current/addresses	

Method	URL	Body Parameters
		Content-Type: application/x-www-
PUT	https://localhost:9002/rest/v1/electronics/cart/address/delivery/8796192014359	
GET	https://localhost:9002/rest/v1/electronics/cart/deliverymodes	
PUT	https://localhost:9002/rest/v1/electronics/cart/deliverymodes/premium-gross	
GET	https://localhost:9002/rest/v1/electronics/customers/current/paymentinfos	
PUT	https://localhost:9002/rest/v1/electronics/cart/paymentinfo/8796158591018	
POST	https://localhost:9002/rest/v1/electronics/cart/authorize	securityCode=123
POST	https://localhost:9002/rest/v1/electronics/cart/placeorder	
GET	https://localhost:9002/rest/v1/electronics/orders	
GET	https://localhost:9002/rest/v1/electronics/orders/00002036	
POST	https://localhost:9002/rest/v1/customers/current/logout	

Method	URL	Body Parameters
		Content-Type: application/x-www-
POST	https://localhost:9002/rest/v1/electronics/cart/paymentinfo	accountHolderName: "John Doe" cardNumber: "4111111111111111" cardType: "visa" expiryMonth: "01" expiryYear: "2017" saved: "true" defaultPaymentInfo: "true" billingAddress.titleCode: "mr" billingAddress.firstName: "John" billingAddress.lastName: "Doe" billingAddress.line1: "Toyosaka" billingAddress.line2: "3-16-19" billingAddress.postalCode: "53" billingAddress.town: "Osaka" billingAddress.country.isocode: "JP" billingAddress.region.isocode: "JP"

Error Responses from OCC v1

Error responses returned by the `ycommercewebservices` extension.

Error Formats

Error responses used in OCC Web services have a unified format that can describe a single error or an error list.

Error Attribute	Mandatory?	Description
<code><type></code>	Yes	Type of error, for example: <code>AccessDeniedError</code> , <code>ValidationError</code> , etc.
<code><message></code>	Yes	Text message that describes the error.
<code><subjectType></code>	No	Subject type, for example: parameter, site
<code><subject></code>	No	Subject identifier, for example: <code><lastName></code> , <code><electronics></code>
<code><reason></code>	No	Error reason code, for example: invalid, missing, <code><unknownIdentifier></code>

Response Formats

XML Error Format

An example of an error in XML format:

```

<errors>
  <error>
    <message>...</message>
    <reason>...</reason>
    <subject>...</subject>
    <subjectType>...</subjectType>
    <type>...</type>
  </error>

  <error>
    ...
  </error>

```

</error>

...

</errors>

JSON Error Format

An example of an error in JSON format:

```
{
  "errors": [
    {
      "message": "...",
      "reason": "...",
      "subject": "...",
      "subjectType": "...",
      "type": "..."
    },
    {
      "message": "...",
      "reason": "...",
      "subject": "...",
      "subjectType": "...",
      "type": "..."
    }
  ]
}
```

Standard Errors

Authentication and Authorization Errors

Response Status	Case	Response Body
HTTP/1.1 401 Unauthorized	<p>When the request requires user authentication or the user has no rights to the resource.</p> <p>Examples:</p> <ul style="list-style-type: none"> GET - https://localhost:9002/rest/v1/wsTest/customers/current/addresses without authorization token GET - https://localhost:9002/rest/v1/wsTest/customers/current/addresses with client authorization token 	<pre><?xml version="1.0" en <errors> <error> <message>Acces <type>AccessDe </error> </errors></pre>
HTTP/1.1 401 Unauthorized	<p>Missing client_id parameter in token request.</p> <p>Example: POST - https://localhost:9002/authorizationserver/oauth/token</p> <p>Header:</p> <p>Content-Type: application/x-www-form-urlencoded</p> <p>Request body:</p> <p>client_secret=secret&grant_type=client_credentials</p>	<pre><?xml version="1.0" en <errors> <error> <message>An Au <type>Unauthor </error> </errors></pre>
HTTP/1.1 401 Unauthorized	<p>Wrong client_id in token request.</p> <p>Example: POST - https://localhost:9002/authorizationserver/oauth/token</p> <p>Header:</p> <p>Content-Type: application/x-www-form-urlencoded</p> <p>Request body:</p> <p>client_id=wrong_client_id&client_secret=secret&grant_type=client_credentials</p>	<pre><?xml version="1.0" en <errors> <error> <message>No cl <type>Unauthor </error> </errors></pre>
HTTP/1.1 401 Unauthorized	<p>Wrong client_secret in token request.</p> <p>Example: POST - https://localhost:9002/authorizationserver/oauth/token</p> <p>Header:</p> <p>Content-Type: application/x-www-form-urlencoded</p>	<pre><?xml version="1.0" en <errors> <error> <message>Bad c <type>BadClie</pre>

Response Status	Case	Response Body
	Request body: <code>client_id=mobile_android&client_secret=wrong_secret&grant_type=client_credentials</code>	<code></error></code> <code></errors></code>
HTTP/1.1 401 Unauthorized	When the authorization token is incorrect.	<code><?xml version="1.0" en</code> <code><errors></code> <code><error></code> <code><message>Inval</code> <code><type>InvalidT</code> <code></error></code> <code></errors></code>
HTTP/1.1 401 Unauthorized	When the token has expired.	<code><?xml version="1.0" en</code> <code><errors></code> <code><error></code> <code><message>Acces</code> <code><type>InvalidT</code> <code></error></code> <code></errors></code>
HTTP/1.1 400 Bad Request	Wrong grant type in the token request. Example: POST - <code>https://localhost:9002/authorizationserver/oauth/token</code> Header: Content-Type: <code>application/x-www-form-urlencoded</code> Request body: <code>client_id=mobile_android&client_secret=secret&grant_type=wrong_credentials</code>	<code><?xml version="1.0" en</code> <code><errors></code> <code><error></code> <code><message>Unsup</code> <code><type>Unsuppor</code> <code></error></code> <code></errors></code>
HTTP/1.1 400 Bad Request	Missing grant type in the token request. Example: POST - <code>https://localhost:9002/authorizationserver/oauth/token</code> Header: Content-Type: <code>application/x-www-form-urlencoded</code> Request body: <code>client_id=mobile_android&client_secret=secret</code>	<code><?xml version="1.0" en</code> <code><errors></code> <code><error></code> <code><message>Missi</code> <code><type>InvalidR</code> <code></error></code> <code></errors></code>
HTTP/1.1 400 Bad Request	Missing or wrong customer credentials in the customer's token request. Example: POST - <code>https://localhost:9002/authorizationserver/oauth/token</code> Header: Content-Type: <code>application/x-www-form-urlencoded</code> Request body: <code>client_id=mobile_android&client_secret=secret&grant_type=password</code>	<code><?xml version="1.0" en</code> <code><errors></code> <code><error></code> <code><message>Bad c</code> <code><type>InvalidG</code> <code></error></code> <code></errors></code>

Bad Request Errors

Response Status	Case	Response Body
HTTP/1.1 400 Bad Request	When there is no required parameter in the request. Example: GET - <code>https://localhost:9002/rest/v1/wsTest/products/export/incremental</code> without required timestamp parameter	<code><?xml version="1.0" encoding="UTF-16"?)></code> <code><errors></code> <code><error></code> <code><message>Required String parameter</code> <code><type>MissingServletRequestParamet</code> <code></error></code> <code></errors></code>

Response Status	Case	Response Body
HTTP/1.1 404 Not Found	When the resource does not exist. Example: GET - https://localhost:9002/rest/v1/electronics/methodDoesNotExist	<pre><?xml version='1.0' encoding='UTF-8'?> <errors> <error> <message>There is no resource for pa <type>UnknownResourceError</type> </error> </errors></pre>
HTTP/1.1 405 Method Not Allowed	When the HTTP method type is not allowed. Example: PUT - https://localhost:9002/rest/v1/electronics/products	<pre><?xml version="1.0" encoding="UTF-16"?> <errors> <error> <message>Request method 'PUT' not <type>HttpRequestMethodNotSupporte </error> </errors></pre>

hybris Specific Errors

General Errors

Response Status	Error	Case
HTTP/1.1 400 Bad Request	ValidationError	<p>General case:</p> <p>When there is no required parameter or a parameter has an incorrect value. Example errors are provided below.</p> <hr/> <p>When there is no parameter or an incorrect parameter in the add address request.</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/customers/current/address</p> <p>Header:</p> <p>Content-Type: application/x-www-form-urlencoded</p> <p>Request body:</p> <p>line1=Toyosaki&line2=3-16-19&town=Osaka&postalCode=531-0072&country.isocode=JP</p> <hr/> <p>When there is no parameter or an incorrect parameter in the add payment information request.</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/cart/paymentinfo</p> <p>Header:</p> <p>Content-Type: application/x-www-form-urlencoded</p> <p>Request body:</p>

Response Status	Error	Case
		<p>accountHolderName=John+Doe&cardNumber=4111111111111111&expiryMonth=01&billingAddress=3-16-19&billingAddress.postalCode=531-0072&billingAddress.town=Osaka&billingAddress</p>
		<p>When there is no parameter or an incorrect parameter in the add review request.</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/products/872912/reviews</p>
		<p>When the storeName parameter is incorrect (for example, the store does not exist) for a pickup in store entry.</p> <p>Example:</p> <ul style="list-style-type: none">• POST -https://localhost:9002/rest/v1/electronics/cart/entry?code=489702&store• PUT - https://localhost:9002/rest/v1/electronics/cart/entry/0/store?storeName=
		<p>When there is no country object with the given code in the database.</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/customers/current/address</p> <p>Header:</p> <p>Content-Type: application/x-www-form-urlencoded</p> <p>Request body:</p> <p>firstName=John&lastName=Doe&titleCode=mr&line1=Toyosaki&line2=3-16-19&town=Osaka&</p>
		<p>When the login parameter has an incorrect value.</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/customers</p> <p>Header:</p> <p>Content-Type: application/x-www-form-urlencoded</p> <p>Request body:</p>

Response Status	Error	Case
		login=WRONG_LOGIN&password=your_password&firstName=John&lastName=Doe&titleCode=mr
		<p>When the promotion type parameter is incorrect.</p> <p>Example: GET - https://localhost:9002/rest/v1/electronics/promotions?type=unknownType</p>
		<p>When the timestamp parameter has an incorrect format.</p> <p>Example: GET - https://localhost:9002/rest/v1/electronics/products/expressUpdate?timestamp=2013-06-21</p>
HTTP/1.1 400 Bad Request	InvalidResourceError	<p>When there is a wrong base site ID in the request.</p> <p>Example: GET - https://localhost:9002/rest/v1/wrongBaseSite/titles</p>
HTTP/1.1 400 Bad Request	AmbiguousIdentifierError	<p>When there is more than one resource with the given ID.</p> <p>Example: GET - https://localhost:9002/rest/v1/electronics/products/ 12345</p>
HTTP/1.1 400 Bad Request	UnknownIdentifierError	<p>General case: When the resource with the given ID does not exist.</p>
		<p>When the product with the given code does not exist.</p> <p>Example: GET - https://localhost:9002/rest/v1/electronics/products/13820867876 and pr</p>
		<p>When the title code is incorrect.</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/customers/current/addresses</p> <p>Header:</p> <p>Content-Type: application/x-www-form-urlencoded</p> <p>Request body:</p>

Response Status	Error	Case
		<p><code>firstName=John&lastName=Doe&titleCode=WRONG_ID&line1=Toyosaki&line2=3-16-19&town=</code></p> <p>When the store with the given ID does not exist.</p> <p>Example: GET - <code>https://localhost:9002/rest/v1/electronics/stores/WRONG_ID</code></p>
HTTP/1.1 400 Bad Request	ModelSavingError	When an error occurs while saving a resource to the database.

Resource Customer Errors

Response Status	Error	Case
HTTP/1.1 400 Bad Request	DuplicateUidError	<p>When a customer cannot be created because the given login already exists.</p> <p>Example: POST - <code>https://localhost:9002/rest/v1/electronics/customers</code></p> <p>Header:</p> <p>Content-Type: <code>application/x-www-form-urlencoded</code></p> <p>Request body:</p> <p><code>login=test1@test.com&password=your_password&firstName=John&lastName=Doe&titleCode=mr</code></p> <p>when customer with login test1@test.com already exists</p>
HTTP/1.1 400 Bad Request	PasswordMismatchError	<p>When the given password is incorrect.</p> <p>Example: PUT - <code>https://localhost:9002/rest/v1/electronics/customers/current/login</code></p> <p>Header:</p> <p>Content-Type: <code>application/x-www-form-urlencoded</code></p> <p>Request body:</p> <p><code>newLogin=aa@bb.com&password=your_password</code></p> <p>and password is incorrect</p>

Resource Cart Errors

Response Status	Error	Case
HTTP/1.1 400 Bad Request	CommerceCartModificationError	<p>General case:</p> <p>This exception is thrown when adding, updating, or removing items from the cart. Error examples are </p> <hr/> <p>When cart entry with the requested number doesn't exist in current cart.</p> <p>Example:</p> <ul style="list-style-type: none"> • PUT - <code>https://localhost:9002/rest/v1/electronics/cart/entry/3?qty=5</code> \ • DELETE - <code>https://localhost:9002/rest/v1/electronics/cart/entry/3</code> where

Response Status	Error	Case
		<p>When the given quantity is incorrect</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/cart/entry?code=4</p>
HTTP/1.1 400 Bad Request	ValidationError	<p>When the cart is invalid in the place order request. It means that there is no required information for the</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/cart/placeorder</p>
HTTP/1.1 400 Bad Request	UnsupportedDeliveryAddressError	<p>When the address set as the delivery address is incorrect; for example: the address does not exist, the</p> <p>Example: PUT - https://localhost:9002/rest/v1/electronics/cart/address/delivery</p>
HTTP/1.1 400 Bad Request	UnsupportedDeliveryModeError	<p>When the delivery mode does not exist or when the delivery address is not set for the cart.</p> <p>Example: PUT - https://localhost:9002/rest/v1/electronics/cart/deliverymode</p>
HTTP/1.1 400 Bad Request	InvalidPaymentInfoError	<p>When there was a problem with setting the payment information for the cart; for example no session cart</p> <p>Example: PUT - https://localhost:9002/rest/v1/electronics/cart/paymentinfo/8</p>
HTTP/1.1 400 Bad Request	PaymentAuthorizationError	<p>When there are problems with the payment authorization; for example: there is no session cart or no p</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/cart/authorize?sc</p>
HTTP/1.1 400 Bad Request	NoCheckoutCartError	<p>General case:</p> <p>This exception is thrown when there is no session cart during a checkout process step. Before the call</p>

Response Status	Error	Case
		<p>When there is no cart in session provided in the set delivery address request.</p> <p>Example: PUT - https://localhost:9002/rest/v1/electronics/cart/address/deli and you forgot to add a JSESSIONID cookie from the previous request</p>
		<p>When there is no cart in session when applying or releasing a voucher.</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/cart/voucher/xyz-</p>
		<p>When there is no cart in session when adding payment information.</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/cart/paymentinfo</p> <p>Header:</p> <p>Content-Type: application/x-www-form-urlencoded</p> <p>Request body:</p> <p>accountHolderName=John+Doe&cardNumber=4111111111111111&cardType=visa&exp.16-19&billingAddress.postalCode=531-0072&billingAddress.town=Osaka&billi</p>
		<p>When there is no cart in session in a place order request.</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/cart/placeorder</p>
HTTP/1.1 400 Bad Request	CommerceCartRestorationError	<p>When the cart that you want to restore does not exist.</p> <p>Example: GET - https://localhost:9002/rest/v1/electronics/cart/restore?guid=</p>
HTTP/1.1 400 Bad Request	InsufficientStockError	<p>General case:</p> <p>When the product is out of stock or there is low stock for it. Error examples are provided below.</p>
		<p>When the product is out of stock when adding it to the cart.</p> <p>Example: POST https://localhost:9002/rest/v1/electronics/ cart/entry?code=</p>
		<p>When the product is out of stock in the store where it is to be picked up.</p> <p>Example: PUT https://localhost:9002/rest/v1/electronics /cart/entry/0/store</p>

Response Status	Error	Case
		<p>When the product is out of stock when trying to place the order.</p> <p>Example: POST https://localhost:9002/rest/v1/electronics/ cart/placeorder</p>
HTTP/1.1 400 Bad Request	StockSystemError	<p>When the stock system is not enabled and there is no information about the stock for the product.</p> <p>Example:</p> <ul style="list-style-type: none"> • POST https://localhost:9002/rest/v1/electronics/cart/entry?code=489 • PUT https://localhost:9002/rest/v1/electronics/cart/entry/store?stc
HTTP/1.1 400 Bad Request	CommercePromotionRestrictionError	<p>When there is no PromotionOrderRestriction for the promotion when we try to apply the promotion to</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/cart/promotion/01</p>
HTTP/1.1 400 Bad Request	VoucherOperationError	<p>When an error occurs during a voucher operation. Error examples are provided below.</p> <p>When trying to apply a non-existent voucher.</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/cart/voucher/notE</p> <p>When an error occurs during the voucher-application process.</p> <p>Example: POST - https://localhost:9002/rest/v1/electronics/cart/voucher/xyz-</p> <p>When an error occurs during the release voucher process.</p> <p>Example: DELETE - https://localhost:9002/rest/v1/electronics/cart/voucher/xyz</p>

Resource Customer Group Errors

Response Status	Error	Case	
HTTP/1.1 400 Bad Request	InvalidCustomerGroupError	When the requested group is not a sub-group of the customergroup. Example: GET - https://localhost:9002/rest/v1/electronics/customergroups/customermanagergroup	R

Voucher Resource Errors

Response Status	Exception	Case	Response Body
HTTP/1.1 400 Bad Request	VoucherOperationError	When the voucher with the given code does not exist. Example: GET - https://localhost:9002/rest/v1/electronics/vouchers/abc-9PSW-EDH2	<?xml version="1.0" encoding="UTF-8"> <errors> <error> <message>Voucher not found</message> <type>VoucherOperationError</type> </error> </errors>

Cross-Origin Resource Sharing

Cross-Origin Resource Sharing (CORS) is a W3C effort to introduce a standard mechanism for enabling cross-domain requests in web browsers and participating servers.

Overview

The SAP Commerce software uses the open source CORS filter, which is a standard Java EE filter.

Cross-Origin Resource Sharing allows web browsers to overcome the same origin policy, which typically prohibits cross-site HTTP requests. CORS is supported by most modern web browsers, including Internet Explorer from version 10 on.

Default Configuration for ycommercewebservices Extension

The web.xml file for the ycommercewebservices extension ships with the default configuration that allows browser-based JavaScript clients to use the RESTful ycommercewebservices APIs.

```
<filter>
  <filter-name>CORS</filter-name>
  <filter-class>com.thetransactioncompany.cors.CORSFilter</filter-class>
  <init-param>
    <param-name>cors.supportedMethods</param-name>
    <param-value>GET, POST, HEAD, PUT, DELETE, OPTIONS</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>CORS</filter-name>
  <url-pattern>*</url-pattern>
</filter-mapping>
```

Once HTTP Request is made by a browser from a different origin, this filter automatically adds the default CORS Headers to the response. The initiating browser needs to understand the CORS Headers in order to allow the cross-site request.

Sample Request

```
GET http://earlybird.hybris.com:9001/rest/v1/electronics/products?query=camera
Origin:
http://nfcbeer Garten.appspot.com
...
```

Sample Responses

```
200 OK
Access-Control-Allow-Credentials:
true
Access-Control-Allow-Origin:
http://nfcbeer Garten.appspot.com
...
```

Disabling CORS

If you do not have a browser-based JavaScript client accessing your API, disable CORS. To do so, remove the filter and filter mapping from the `web.xml` file. Additionally, remove the CORS jar file from `\ext-hybris\ycommercewebservices\web\webroot\WEB-INF\lib` directory.

Upgrading to OCC Extensions

Guidelines on upgrading from v1 to Extensions-based implementation of the OCC API.

Modified URLs

You must change URLs to point to new resources. The major change is the version number in the URL that is now v2. All resource URL paths must also be adapted to the new convention. The specific resource mapping is described in the [API resources mapping](#) below.

New Return Types

In OCC Extensions, all returned objects have been changed to WsDTO objects, so you must change all response processing to the new response structure. All WsDTO objects returned by the API methods are described in the [WsDTO Concept](#) page.

New Request Parameters

In OCC Extensions, methods have a new request parameter that specifies the form of the returned object. This request parameter is provided by default. However, to ensure that the response objects provide all the fields that you require, you can customize the request parameter.

API Resource Mapping

The following table shows you the API resource mapping.

V1		OCC Extensions
<code>/ {site} /customers</code>	POST	<code>/ {site} /users</code>
<code>/ {site} /customers/current</code>	GET	<code>/ {site} /users/{userId}</code>
<code>/ {site} /customers/current/login</code>	PUT	<code>/ {site} /users/{userId}/login</code>
<code>/ {site} /customers/current/profile</code>	PUT	<code>/ {site} /users/{userId}</code>
<code>/ {site} /customers/current/password</code>	PUT,POST	<code>/ {site} /users/{userId}/password</code>
<code>/ {site} /customers/current/forgottenpassword</code>	POST	<code>/ {site} /forgottenpasswordtokens</code>
<code>/ {site} /customers/current/addresses</code>	GET	<code>/ {site} /users/{userId}/addresses</code>

V1		OCC Extensions
/{{site}}/customers/current/addresses	POST	/{{site}}/users/{userId}/addresses
/{{site}}/customers/current/addresses/{id}	PUT	/{{site}}/users/{userId}/addresses/{id}
/{{site}}/customers/current/addresses/{id}	DELETE	/{{site}}/users/{userId}/addresses/{id}
/{{site}}/customers/current/addresses/default/{id}	PUT	removed - use /{{site}}/users/{userId}/addresses/default/{id}
/{{site}}/customers/current/paymentinfos	GET	/{{site}}/users/{userId}/paymentdetails
/{{site}}/customers/current/paymentinfos/{id}	GET	/{{site}}/users/{userId}/paymentdetails/{id}
/{{site}}/customers/current/paymentinfos/{id}	DELETE	/{{site}}/users/{userId}/paymentdetails/{id}
/{{site}}/customers/current/paymentinfos/{id}	PUT	/{{site}}/users/{userId}/paymentdetails/{id}
/{{site}}/customers/current/paymentinfos/{id}/address	PUT	/{{site}}/users/{userId}/paymentdetails/{id}/address
/{{site}}/customers/current/customergroups	GET	/{{site}}/users/{userId}/customergroups
/{{site}}/customers/{uid}/customergroups	GET	/{{site}}/users/{userId}/customergroups/{uid}
/{{site}}/customers/current/locationLatLong	PUT	removed - because there is no session
/{{site}}/customers/current/location	PUT	removed - because there is no session
/{{site}}/customers/current/location	GET	removed - because there is no session
/{{site}}/customer/current/guestlogin	POST	/{{site}}/users/{userId}/carts/{cartId}/guestlogin
/{{site}}/customer/current/convert	POST	/{{site}}/users/{userId}/convert
/{{site}}/orders	GET	/{{site}}/users/{userId}/orders
/{{site}}/orders/{code}	GET	/{{site}}/users/{userId}/orders/{code}
CARTS		
/{{site}}/cart	GET	/{{site}}/users/{userId}/carts/{cartId}
/{{site}}/cart/entry	POST	/{{site}}/users/{userId}/carts/{cartId}/entry
/{{site}}/cart/entry/{entryNumber}	PUT,DELETE	/{{site}}/users/{userId}/carts/{cartId}/entry/{entryNumber}
/{{site}}/cart/address/delivery/{id}	PUT	/{{site}}/users/{userId}/carts/{cartId}/address/delivery/{id}
/{{site}}/cart/address/delivery	DELETE	/{{site}}/users/{userId}/carts/{cartId}/address/delivery
/{{site}}/cart/deliverymodes/{code}	PUT	/{{site}}/users/{userId}/carts/{cartId}/deliverymodes/{code}
/{{site}}/cart/deliverymodes	GET,DELETE	/{{site}}/users/{userId}/carts/{cartId}/deliverymodes
/{{site}}/cart/placeorder	POST	/{{site}}/users/{userId}/orders?cartId={cartId}&securityCode={securityCode}
/{{site}}/cart/paymentinfo	POST	/{{site}}/users/{userId}/carts/{cartId}/paymentinfo
/{{site}}/cart/paymentinfo/{id}	PUT	/{{site}}/users/{userId}/carts/{cartId}/paymentinfo/{id}
/{{site}}/cart/authorize	POST	removed
/{{site}}/cart/restore	GET	removed
/{{site}}/cart/promotion/{promotionCode}	POST	/{{site}}/users/{userId}/carts/{cartId}/promotion/{promotionCode}
/{{site}}/cart/promotion/{promotionCode}	DELETE	/{{site}}/users/{userId}/carts/{cartId}/promotion/{promotionCode}
/{{site}}/cart/voucher/{voucherCode}	POST	/{{site}}/users/{userId}/carts/{cartId}/voucher/{voucherCode}

V1		OCC Extensions
/ {site} /cart/voucher/{voucherCode}	DELETE	/ {site} /users/{userId}/carts/{cartId}
/ {site} /cart/checkout	POST	
CATALOGS		
/ {site} /catalogs	GET	/ {site} /catalogs
/ {site} /catalogs/{id}	GET	/ {site} /catalogs/{catalogId}
/ {site} /catalogs/{catalogId}/{catalogVersionId}	GET	/ {site} /catalogs/{catalogId}/{catalogVersionId}
/ {site} /catalogs/{catalogId}/{catalogVersionId}/categories/{category}	GET	/ {site} /catalogs/{catalogId}/{catalogVersionId}/categories/{category}
CUSTOMERGROUPS		
/ {site} /customergroups	GET,POST	/ {site} /customergroups
/ {site} /customergroups/{uid}/members	PUT	/ {site} /customergroups/{groupId}/members
/ {site} /customergroups/{uid}/members/{userId}	DELETE	/ {site} /customergroups/{groupId}/members/{userId}
/ {site} /customergroups/{uid}	GET	/ {site} /customergroups/{groupId}
ORDERS		
/ {site} /orders/byGuid/{guid}	GET	/ {site} /orders/{code}
/ {site} /orders/statusFeed	GET	/ {site} /feeds/orders/statusfeed
PROMOTIONS		
/ {site} /promotions	GET	/ {site} /promotions
/ {site} /promotions/{code}	GET	/ {site} /promotions/{code}
MISC		
/ {site} /languages	GET	/ {site} /languages
/ {site} /currencies	GET	/ {site} /currencies
/ {site} /deliverycountries	GET	/ {site} /deliverycountries
/ {site} /titles	GET	/ {site} /titles

V1		OCC Extensions
/ {site} /cardtypes	GET	/ {site} /cardtypes
STORES		
/ {site} /stores	GET	/ {site} /stores
/ {site} /stores/ {name}	GET	/ {site} /stores/ {storeId}
VOUCHERS		
/ {site} /vouchers/ {code}	GET	
PRODUCTS		
/ {site} /products	GET	/ {site} /products/search
/ {site} /products {productCode} /stock?storeName=<name>	GET	/ {site} /products/ {productCode} /sto
/ {site} /products/export/full	GET	/ {site} /products/export/full
/ {site} /products/export/incremental	GET	/ {site} /products/export/incremental
/ {site} /products/export/references/ {code}	GET	/ {site} /products/ {productCode} /refe
/ {site} /products/expressUpdate	GET	/ {site} /products/expressUpdate
/ {site} /products/ {code}	GET	/ {site} /products/ {productCode}
/ {site} /products/suggest	GET	/ {site} /products/suggestions
/ {site} /products/ {code} /reviews	POST	/ {site} /products/ {productCode} /rev:
/ {site} /products/ {code} /nearLocation	GET	/ {site} /products/ {productCode} /sto
/ {site} /products/ {code} /nearLatLong	GET	/ {site} /products/ {productCode} /sto <latitude>