Seller API Documentation

TABLE OF CONTENTS

USER GUIDE

Getting started

The Daraz Seller Center platform provides a strong tech foundation to help implement impactful features in the future. This is in line with the goal to provide better services and experience to Daraz sellers.

The Seller Center enables sellers to manage products and orders in their online stores, and the Seller Center API is also available to support programmatic maintenance of products and orders.

This guide is targeted at:

- Existing integrated sellers (with Seller Center API) for smooth migration.
- Sellers who are starting an integration project with Daraz Seller Center API.

Most of the API endpoints are backward compatible, albeit some changes to the Product API endpoints and the endpoint URLs. Note that some endpoints are no longer available in the new Seller Center.

Overview of the Seller Center API

The Seller Center enables sellers to manage products and orders in their online stores, and the Seller Center API supports programmatic maintenance of products, orders, package shipment, and finance.
This guide is targeted at:

- Existing integrated sellers (with Seller Center API) for smooth migration.
- Sellers who are starting an integration project with Daraz Seller Center API.

API endpoints

The current Seller Center API provides the following endpoints:

- Product Endpoints
- Sales Order Endpoints
- Quality Control Endpoints
- Seller Endpoints

Detailed description of all the API calls is provided in the sections below.

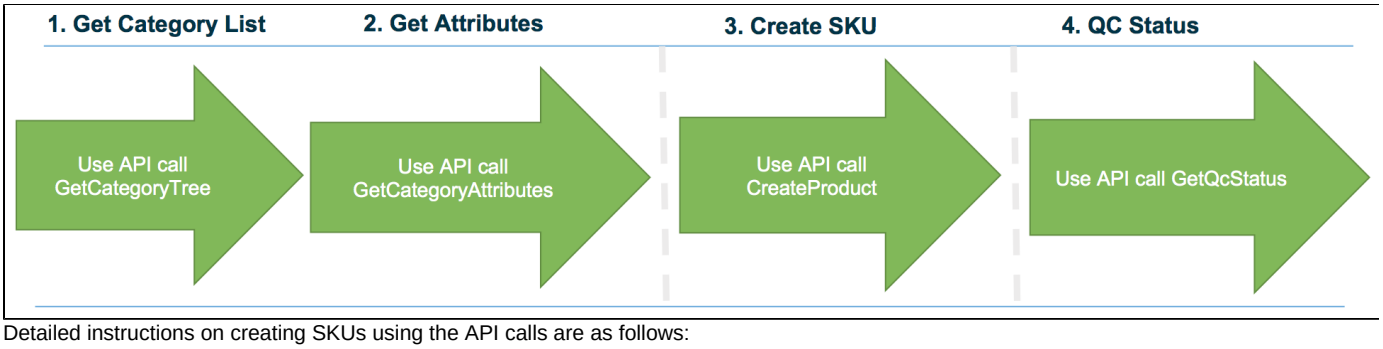
Process flow

With the API calls, you can manage products, process orders and shipment, and view metrics and statistics. The detailed process flow is as follows.

Product listing

The product management flow enables users to use API calls to search for categories, category attributes, and product brand for the SKUs to be created. With the returned information, sellers fill in the category, category attributes, and other product information to create the product.

The product listing process and steps are as follows:



Detailed instructions on creating SKUs using the API calls are as follows:

Step 1: Use *GetCategoryTree* to search for the appropriate category ID for the SKUs to be created.

Step 2: Use *GetCategoryAttributes* to retrieve the attributes of the specified category. The category ID and category attributes will be used as reference to create SKUs.

Step 3: Use *UploadImage*/*UploadImage* or *MigrateImage*/*MigrateImage* to get the URL of images for the SKUs to be created.

Step 4: Based on the selected category, category attributes, image URLs, and other product information, create SKUs. Detailed steps are as follows:

- In the <*PrimaryCategory*> tag, enter the category ID for the SKUs.
- (Optional) Use the <*AssociatedSku*> tag if you want to add SKUs to an existing product.
- In the <*Skus*> tag, enter the product information for the SKU. Multiple SKUs can be added.

Step 5: Use *GetQcStatus* to download QC status of the product listing.

Image, price, and quantity maintenance

The image, price and quantity maintenance process and steps are as follows.

Step 1: Use *UploadImage*/*UploadImages* to upload an image or images of the product from local device to the Daraz server.

Step 2: Use *MigrateImage*/*MigrateImages* to retrieve the image URL from an external site to the Daraz server. Note that only valid image URLs from Daraz servers can be used for *CreateProduct* and *SetImage*.

Step 3: Use *SetImage* for existing SKUs by associating one or more image URLs within the same request XML.

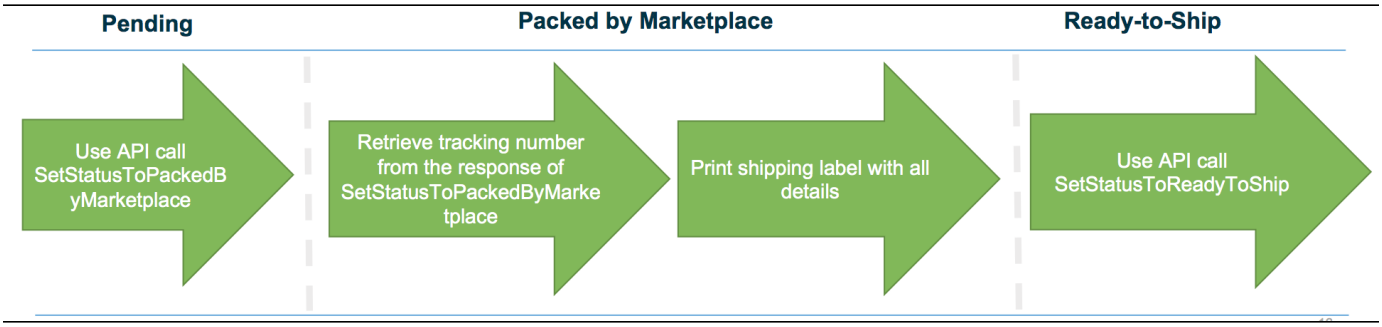
Step 4: Use *GetProducts* to retrieve the latest product information under seller center account, which can be filtered by all/active/inactive/deleted etc.

Step 5: Use *UpdatePriceQuantity* to update price and quantity of respective products as necessary. The maximum number that can be updated at once is 20.

Order processing and shipping

Order related API calls are backward compatible, meaning that the call structures and behavior remain the same.

The order processing and shipping steps are as follows:



Detailed instructions on order processing with the API calls are as follows:

Step 1: Use *GetOrder* to retrieve order information. Based on the selected category, category attributes, image URLs, and other product information, create SKUs. Detailed steps are as follows:

- Use the *UpdatedAfter* timestamp to filter out orders retrieved.
- Use *Limit* and *Offset* parameters to set the number of orders returned.
- Save the order IDs.

Step 2: Use *GetOrderItems* to retrieve the item information within an order.

Step 3: Use *GetShipmentProvider* to retrieve 3rd party shipment provider (3PL) list. Ignore this call if 3PL gets automatically assigned to you.

Step 4: Use *SetStatustoPackedByMarketplace* to update the status of the specific order by providing the order item ID, and shipment provider (send null in case of automatic assignment). For multiple items in the same package, input item IDs commas separated enclosed by square brackets. The seller center will respond to single package tracking number.

Step 5: Use *GetDocuments* to get the template for the air waybill, invoice, and pick list.

Step 6: Use *SetStatustoReadyToShip* to update the order status to "Ready to Ship" (RTS).

Metrics, Payout Status, Statistics, and Transaction details

Use the following seller endpoints to retrieve metrics, payout status, statistics, and transaction details of the store.

Step 1: Use *GetMetrics* to retrieve sales and order metrics for a specified period (one day, week, month, or all time). Sales and order metrics include the number of total/active SKUs and orders, and the amount of sales and commission.

Step 2: Use *GetPayoutStatus* to retrieve payout amount and status for a specified period. Returned payout information includes opening balance, item revenue, shipping fee, shipping fee credit, refunds, closing balance, and guarantee deposit. The payout status is 1 (paid) or 0 (not paid).

Step 3: Use *GetStatistics* to retrieve product and order statistics for a specified period, including:

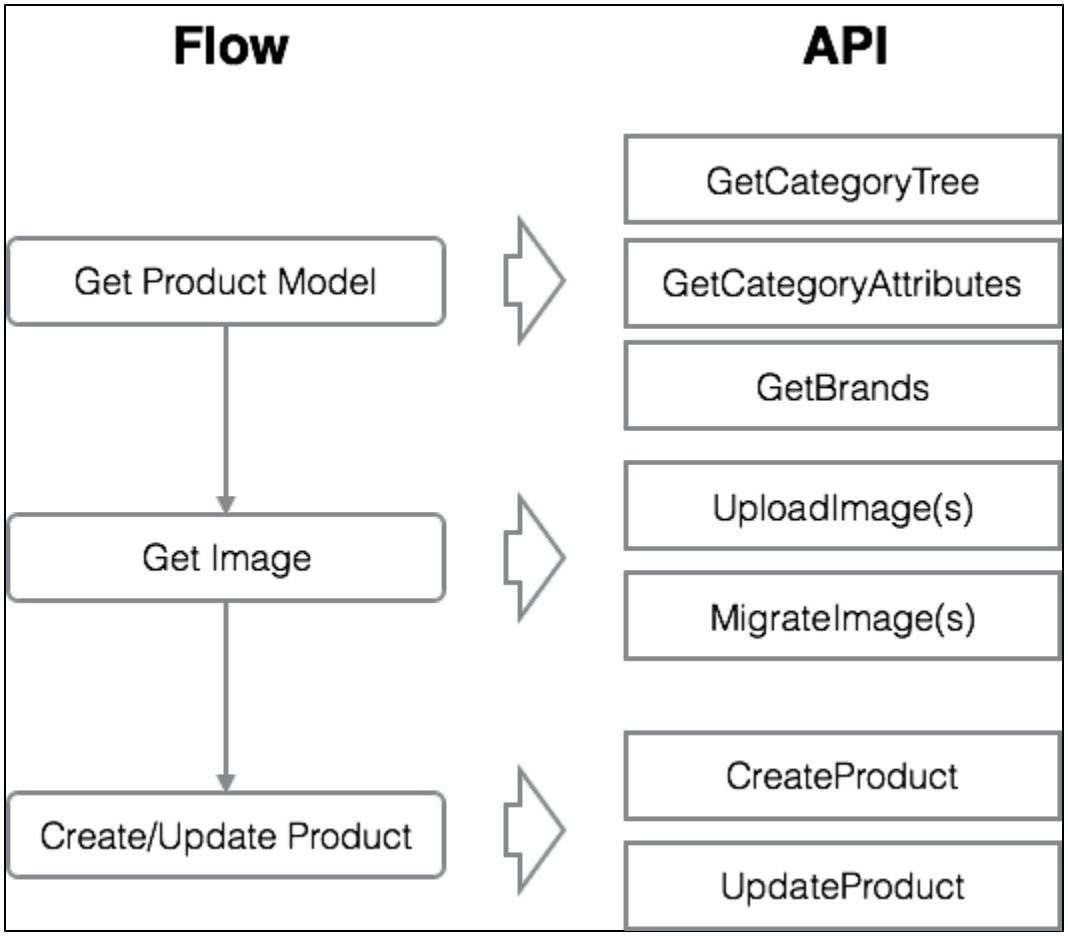
- Products - the number of products in various statuses.
- Orders - the number of orders in various statuses and the total order amount.
- Order items pending - the number of pending order items in the last 24 hours, 48 hours, and earlier.
- Account health - the percentage of orders shipped within 48 hours and percentage of cancelled orders.

Step 4: Use *GetTransactionDetails* to retrieve transaction or fee details for a specified period. Transaction details include transaction date, type, number, amount, statement, payment status, order number, order item number, shipment type, shipment provider, and reference ID.

Guide for Creating Products

Process flow and related APIs

The following diagram shows the process flow and related APIs for creating and updating products.



Process flow

Step 1: Get product model information through the category tree and category attributes, for example, which attributes (fields) are mandatory for a product, and what content or options should be provided for the attributes.

Step 2: Retrieve the URL of images that are required for creating or updating a product.

Step 3: Create a product or update the information of a listed product.

Related APIs

The following APIs will be used to create a product:

1. Use *GetCategoryTree* API to read the category tree. Determine the appropriate category for the product to be created. Each category has its own attributes required for product creation.

2. Use *GetCategoryAttributes* with the category ID to retrieve the attributes of the category. The category ID and category attributes will be used as reference to create the product. Note that the "brand" attribute needs to be retrieved separately with the *GetBrands* API.
3. Use the *UploadImage* or *MigrateImage* API to load your product images to Daraz image repository and get the image URL.
4. Based on the selected category, category attributes, image URLs, and other product information, you can create the product through the *CreateProduct* API.

Usage demo

Step 1: Get the category tree, and then determine the leaf category for the product.

XML


```
<?xml version="1.0" encoding="utf-8"?>
<SuccessResponse>
    <Head>
        <RequestId></RequestId>
        <RequestAction>GetCategoryTree</RequestAction>
        <ResponseType>Categories</ResponseType>
        <Timestamp>2018-03-19T15:51:41+08:00</Timestamp>
    </Head>
    <Body>
        <Category>
            <name>Bags and Travel</name>
            <categoryId>1902</categoryId>
            <children>
                <Category>
                    <name>Kids Bags</name>
                    <categoryId>10001930</categoryId>
                    <children>
                        <Category>
                            <name>Accessories</name>
                            <categoryId>10001958</categoryId>
                            <var>false</var>
                            <leaf>true</leaf>
                        </Category>
                        <Category>
                            <name>Backpacks</name>
                            <categoryId>10001957</categoryId>
                            <var>false</var>
                            <leaf>true</leaf>
                        </Category>
                        <Category>
                            <name>Backpacks Trolley</name>
                            <categoryId>10001956</categoryId>
                            <var>false</var>
                            <leaf>true</leaf>
                        </Category>
                        <Category>
                            <name>Bags</name>
                            <categoryId>10001955</categoryId>
                            <var>false</var>
                            <leaf>true</leaf>
```

```
            </Category>
        </children>
        <var>false</var>
        <leaf>false</leaf>
    </Category>
</children>
<var>false</var>
<leaf>false</leaf>
```

```
        </Category>
    </Body>
</SuccessResponse>
```

In the above sample, the field "leaf=true" means that the category is a leaf category, which can be used for creating the product.

Step 2: Retrieve the attributes for the selected category. For example, if the selected category is "Bags and Travel > Kids Bags > Bags", then "Bags" is the leaf category, with the category ID "10001955". Use "PrimaryCategory=10001955" as the input parameter of the *GetCategoryAttributes* API. You will get the following response.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
    <Head>
        <RequestId></RequestId>
        <RequestAction>GetCategoryAttributes</RequestAction>
        <ResponseType>Attributes</ResponseType>
        <Timestamp>2018-03-19T15:59:20+08:00</Timestamp>
    </Head>
    <Body>
        <Attribute>
            <label>Name</label>
            <name>name</name>
            <isMandatory>1</isMandatory>
            <isSaleProp>0</isSaleProp>
            <attributeType>normal</attributeType>
            <inputType>text</inputType>
            <options/>
        </Attribute>
        <Attribute>
            <label>Highlights</label>
            <name>short_description</name>
            <isMandatory>1</isMandatory>
            <isSaleProp>0</isSaleProp>
            <attributeType>normal</attributeType>
            <inputType>richText</inputType>
            <options/>
        </Attribute>
        <Attribute>
            <label>Product Description</label>
            <name>description</name>
            <isMandatory>0</isMandatory>
            <isSaleProp>0</isSaleProp>
            <attributeType>normal</attributeType>
            <inputType>richText</inputType>
            <options/>
        </Attribute>
        <Attribute>
            <label>Brand</label>
```

```
        <name>brand</name>
        <isMandatory>1</isMandatory>
        <isSaleProp>0</isSaleProp>
        <attributeType>normal</attributeType>
        <inputType>singleSelect</inputType>
        <options/>
    </Attribute>
    <Attribute>
        <label>Model</label>
        <name>model</name>
        <isMandatory>1</isMandatory>
        <isSaleProp>0</isSaleProp>
        <attributeType>normal</attributeType>
        <inputType>text</inputType>
        <options/>
    </Attribute>
    <Attribute>
        <label>Color Family</label>
        <name>color_family</name>
        <isMandatory>1</isMandatory>
        <isSaleProp>1</isSaleProp>
        <attributeType>sku</attributeType>
        <inputType>multiSelect</inputType>
        <options>
            <Option>
                <name>Black</name>
            </Option>
            <Option>
                <name>Beige</name>
            </Option>
            <Option>
                <name>...</name>
            </Option>
        </options>
    </Attribute>
    <Attribute>
        <label>Warranty Type</label>
        <name>warranty_type</name>
        <isMandatory>0</isMandatory>
        <isSaleProp>0</isSaleProp>
        <attributeType>normal</attributeType>
        <inputType>singleSelect</inputType>
        <options>
            <Option>
                <name>Local manufacturer warranty</name>
            </Option>
            <Option>
                <name>International Manufacturer Warranty</name>
            </Option>
           <Option>
```

```
            <name>...</name>
        </Option>
    </options>
</Attribute>
<Attribute>
    <label>What's in the box</label>
    <name>package_content</name>
    <isMandatory>0</isMandatory>
    <isSaleProp>0</isSaleProp>
    <attributeType>sku</attributeType>
    <inputType>text</inputType>
    <options/>
</Attribute>
<Attribute>
    <label>Warranty Period</label>
    <name>warranty</name>
    <isMandatory>0</isMandatory>
    <isSaleProp>0</isSaleProp>
    <attributeType>normal</attributeType>
    <inputType>singleSelect</inputType>
    <options>
        <Option>
            <name>1 Month</name>
        </Option>
        <Option>
            <name>2 Months</name>
        </Option>
        <Option>
            <name>3 Months</name>
        </Option>
        <Option>
            <name>4 Months</name>
        </Option>
        <Option>
            <name>5 Months</name>
        </Option>
        <Option>
            <name>...</name>
        </Option>
    </options>
</Attribute>
<Attribute>
    <label>Warranty policy (optional)</label>
    <name>product_warranty</name>
    <isMandatory>0</isMandatory>
    <isSaleProp>0</isSaleProp>
    <attributeType>normal</attributeType>
    <inputType>text</inputType>
    <options/>
</Attribute>
```

```
<Attribute>
    <label>SellerSKU</label>
    <name>SellerSku</name>
    <isMandatory>1</isMandatory>
    <isSaleProp>0</isSaleProp>
    <attributeType>sku</attributeType>
    <inputType>text</inputType>
    <options/>
</Attribute>
<Attribute>
    <label>Barcode</label>
    <name>barcode_ean</name>
    <isMandatory>0</isMandatory>
    <isSaleProp>0</isSaleProp>
    <attributeType>sku</attributeType>
    <inputType>text</inputType>
    <options/>
</Attribute>
<Attribute>
    <label>Quantity</label>
    <name>quantity</name>
    <isMandatory>0</isMandatory>
    <isSaleProp>0</isSaleProp>
    <attributeType>sku</attributeType>
    <inputType>numeric</inputType>
    <options/>
</Attribute>
<Attribute>
    <label>Price (calculated)</label>
    <name>price</name>
    <isMandatory>1</isMandatory>
    <isSaleProp>0</isSaleProp>
    <attributeType>sku</attributeType>
    <inputType>numeric</inputType>
    <options/>
</Attribute>
<Attribute>
    <label>Special Price</label>
    <name>special_price</name>
    <isMandatory>0</isMandatory>
    <isSaleProp>0</isSaleProp>
    <attributeType>sku</attributeType>
    <inputType>numeric</inputType>
    <options/>
</Attribute>
<Attribute>
    <label>Start date of promotion</label>
    <name>special_from_date</name>
    <isMandatory>0</isMandatory>
    <isSaleProp>0</isSaleProp>
```

```
        <attributeType>sku</attributeType>
        <inputType>date</inputType>
        <options/>
    </Attribute>
    <Attribute>
        <label>End date of promotion</label>
        <name>special_to_date</name>
        <isMandatory>0</isMandatory>
        <isSaleProp>0</isSaleProp>
        <attributeType>sku</attributeType>
        <inputType>date</inputType>
        <options/>
    </Attribute>
    <Attribute>
        <label>Images</label>
        <name>__images__</name>
        <isMandatory>0</isMandatory>
        <isSaleProp>0</isSaleProp>
        <attributeType>sku</attributeType>
        <inputType>img</inputType>
        <options/>
    </Attribute>
    <Attribute>
        <label>Taxes</label>
        <name>tax_class</name>
        <isMandatory>0</isMandatory>
        <isSaleProp>0</isSaleProp>
        <attributeType>sku</attributeType>
        <inputType>singleSelect</inputType>
        <options>
            <Option>
                <name>default</name>
            </Option>
        </options>
    </Attribute>
```

```
        <Attribute>...</Attribute>
    </Body>
</SuccessResponse>
```

In the above response:

- "name" represents the name of the attribute that can be recognized by system.
- "label" represents the name of the attribute that will be displayed in the product description page.
- "isMandatory" represents whether the attribute is mandatory or not.
- "isSaleProp" represents whether the attribute is sale property or not. Sale properties are always mandatory.
- "inputType" represents the type of input data, including text, richText, multiSelect, singleSelect, numeric, and date. The entered data must comply with the required data type.

Step 3: Get the brand list by using the *GetBrands* API, and then select the product brand.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId>0a1430201483626042553178 2e</RequestId>
    <RequestAction>GetBrands</RequestAction>
    <ResponseType>Brands</ResponseType>
    <Timestamp>2017-01-05T14:20:42+00:00</Timestamp>
  </Head>
  <Body>
    <Brands>
      <Brand>
        <BrandId>23886</BrandId>
        <Name>2K Games</Name>
        <GlobalIdentifier>2k_games</GlobalIdentifier>
      </Brand>
      <Brand>
        <BrandId>23887</BrandId>
        <Name>3D Crystal Puzzle</Name>
        <GlobalIdentifier>3d_crystal_puzzle</GlobalIdentifier>
      </Brand>
      <Brand>...</Brand>
  </Body>
</SuccessResponse>
```

Step 4: Get the URL of product images by using the *UploadImage(s)* or *MigrateImage(s)* API. Taking the *MigrateImage(s)* API as example:

Input:

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request>
    <Image>

<Url>https://static.wixstatic.com/media/03aa4c_dee4d88cad6340c7b65f3f320
819f130~mv2.gif</Url>
    </Image>
</Request>
```

Output:

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId>0bb606c315214471962122131e2059</RequestId>
    <RequestAction>MigrateImage</RequestAction>
    <ResponseType>Image</ResponseType>
    <Timestamp>2018-03-19T08:13:16+0000</Timestamp>
  </Head>
  <Body>
    <Image>

<Url>https://sg-live.slatic.net/original/b731a8098df7d606ab2e56efc650afc
b.jpg</Url>
      <Code>b731a8098df7d606ab2e56efc650afcb</Code>
    </Image>
  </Body>
</SuccessResponse>
```

Step 5: Use the *CreateProduct* API to create the product, with the category ID, attributes, and attribute values (including brands and images). The data structure of the *UpdateProduct* API is similar with that of the *CreateProduct* API. See the following sample.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<Request>
    <Product>
        <PrimaryCategory>10001958</PrimaryCategory>
        <Attributes>
            <name>test product creatation</name>
            <short_description>test product
```

```
highlights</short_description>
            <description>test product description</description>
            <video />
            <brand>Kid Basix</brand>
            <model>test model</model>
            <recommended_gender>Men</recommended_gender>
            <material>Cotton</material>
            <waterproof>Waterproof</waterproof>
            <warranty_type>Local manufacturer warranty</warranty_type>
            <warranty>1 Month</warranty>
            <product_warranty />
            <Hazmat>Battery,Flammable</Hazmat>
        </Attributes>
        <Skus>
            <Sku>
                <quantity>1</quantity>
                <SellerSku>test-product-creatation-for-api</SellerSku>

                <color_family>Black</color_family>
                <special_price>0.0</special_price>
                <price>23.0</price>
                <package_length>1</package_length>
                <package_weight>1</package_weight>
                <package_content>test what's in the
box</package_content>
                <package_width>1</package_width>
                <package_height>1</package_height>
                <Images>

<Image>https://sg-live.slatic.net/original/b731a8098df7d606ab2e56efc650a
fcb.jpg</Image>
                    <Image />
                    <Image />
                    <Image />
                    <Image />
                    <Image />
                    <Image />
                    <Image />
                </Images>
            </Sku>
```

```
        </Skus>
    </Product>
</Request>
```

Notes:

- Only leaf categories (the deepest level) can be used to create products. Otherwise, the "Category is not leaf" error will be reported.
- The category ID must be retrieved with the *GetCategoryTree* API. If an invalid category ID is used, the "CATEGORY_ID_INVALID" error will be reported.
- The attribute values must comply with the required data format or must be valid options provided by the *GetCategoryAttributes* API. Otherwise, the "xxx is not a valid value" error will be reported.

Appendix. Supported and unsupported HTML tags in the "description" field

The following HTML tags are supported in the "description" field:

XML

```
<tag name="a"/>
<tag name="hr"/>
<tag name="h1"/>
<tag name="h3"/>
<tag name="h2"/>
<tag name="h4"/>
<tag name="h5"/>
<tag name="h6"/>
<tag name="font"/>
<tag name="b"/>
<tag name="i"/>
<tag name="u"/>
<tag name="sup"/>
<tag name="sub"/>
<tag name="strike"/>
<tag name="strong"/>
<tag name="em"/>
<tag name="p"/>
<tag name="br"/>
<tag name="ol"/>
<tag name="li"/>
<tag name="ul"/>
<tag name="div"/>
<tag name="span"/>
<tag name="img"/>
<tag name="map"/>
<tag name="area"/>
<tag name="marquee"/>
<tag name="table"/>
<tag name="tr"/>
<tag name="td"/>
<tag name="caption"/>
<tag name="bgsound"/>
```

```
<tag name="blockquote"/>
<tag name="cite" />
<tag name="small"/>
<tag name="big"/>
<tag name="nobr"/>
<tag name="center"/>
<tag name="dl"/>
<tag name="dt"/>
<tag name="dd"/>
<tag name="pre"/>
<tag name="listing"/>
<tag name="blink"/>
<tag name="spacer"/>
<tag name="th"/>
<tag name="thead"/>
<tag name="tbody"/>
```

```
<tag name="tfoot"/>
<tag name="colgroup"/>
<tag name="col"/>
```

The supported attributes of the tags are as follows:

XML

```
<tag-attributes>
        <attributes name="common">
            <attribute name="style" />
            <attribute name="align" />
            <attribute name="valign" />
            <attribute name="bgcolor" />
            <attribute name="background">
                <regexp-list>
                    <regexp name="offsiteURL" />
                </regexp-list>
            </attribute>
            <attribute name="title" />
            <attribute name="id" />
            <attribute name="class" />
        </attributes>
        <attributes name="style">
            <attribute name="type" />
        </attributes>
        <attributes name="div">
            <attribute name="style" />
            <attribute name="align" />
            <attribute name="valign" />
            <attribute name="bgcolor" />
            <attribute name="background">
                <regexp-list>
                    <regexp name="offsiteURL" />
                </regexp-list>
            </attribute>
            <attribute name="title" />
        </attributes>
        <attributes name="img">
            <attribute name="style" />
            <attribute name="align" />
            <attribute name="valign" />
            <attribute name="bgcolor" />
            <attribute name="background" />
            <attribute name="title" />
            <attribute name="src">
            </attribute>
            <attribute name="border" />
```

```
        <attribute name="width" />
        <attribute name="height" />
        <attribute name="alt" />
        <attribute name="usemap" />
        <!-- add -->
        <attribute name="hspace" />
        <attribute name="ismap" />
        <attribute name="vspace" />
    </attributes>
    <attributes name="font">
        <attribute name="style" />
        <attribute name="align" />
        <attribute name="valign" />
        <attribute name="bgcolor" />
        <attribute name="background">
            <regexp-list>
                <regexp name="offsiteURL" />
            </regexp-list>
        </attribute>
        <attribute name="title" />
        <attribute name="color" />
        <attribute name="size" />
        <attribute name="face" />
    </attributes>
    <attributes name="table">
        <attribute name="style" />
        <attribute name="align" />
        <attribute name="valign" />
        <attribute name="bgcolor" />
        <attribute name="background">
            <regexp-list>
                <regexp name="offsiteURL" />
            </regexp-list>
        </attribute>
        <attribute name="title" />
        <attribute name="border" />
        <attribute name="width" />
        <attribute name="height" />
        <attribute name="cellpadding" />
        <attribute name="cellspacing" />
        <attribute name="bordercolor" />
        <attribute name="blockquote" />
        <!-- add -->
        <attribute name="summary" />
        <attribute name="rules" />
    </attributes>
    <attributes name="td">
        <attribute name="style" />
        <attribute name="align" />
        <attribute name="valign" />
```

```
        <attribute name="bgcolor" />
        <attribute name="background">
            <regexp-list>
                <regexp name="offsiteURL" />
            </regexp-list>
        </attribute>
        <attribute name="title" />
        <attribute name="width" />
        <attribute name="height" />
        <attribute name="colspan" />
        <attribute name="rowspan" />
        <!-- add -->
        <attribute name="headers" />
        <attribute name="scope" />
    </attributes>
    <attributes name="marquee">
        <attribute name="style" />
        <attribute name="align" />
        <attribute name="valign" />
        <attribute name="bgcolor" />
        <attribute name="background">
            <regexp-list>
                <regexp name="offsiteURL" />
            </regexp-list>
        </attribute>
        <attribute name="title" />
        <attribute name="scrollamount" />
        <attribute name="direction" />
        <attribute name="behavior" />
        <attribute name="width" />
        <attribute name="height" />
        <attribute name="scrolldelay" />
        <!-- add -->
        <attribute name="loop" />
    </attributes>
    <attributes name="a">
        <attribute name="style" />
        <attribute name="align" />
        <attribute name="valign" />
        <attribute name="bgcolor" />
        <attribute name="background">
            <regexp-list>
                <regexp name="offsiteURL" />
            </regexp-list>
        </attribute>
        <attribute name="title" />
        <attribute name="target" />
        <attribute name="name" />
        <attribute name="href">
            <regexp-list>
```

```
            <regexp name="offsiteURL" />
        </regexp-list>
    </attribute>
    <!-- add -->
    <attribute name="charset" />
    <attribute name="hreflang" />
    <attribute name="type" />
    <attribute name="shape" />
</attributes>
<attributes name="bgsound">
    <attribute name="src">
        <regexp-list>
            <regexp name="offsiteURL" />
        </regexp-list>
    </attribute>
    <attribute name="loop" />
    <!-- add -->
    <attribute name="autostart" />
</attributes>
<attributes name="map">
    <attribute name="name" />
    <!-- add -->
    <attribute name="title" />
    <attribute name="style" />
</attributes>
<attributes name="area">
    <attribute name="shape" />
    <attribute name="coords" />
    <!-- add -->
    <attribute name="href">
        <regexp-list>
            <regexp name="offsiteURL" />
        </regexp-list>
    </attribute>
    <attribute name="title" />
    <attribute name="style" />
    <attribute name="alt" />
    <attribute name="nohref" />
    <attribute name="target" />
</attributes>
<attributes name="spacer">
    <attribute name="align" />
    <attribute name="valign" />
    <attribute name="type" />
    <attribute name="width" />
    <attribute name="height" />
    <attribute name="size" />
</attributes>
<attributes name="colgroup">
    <attribute name="align" />
```

```
<attribute name="char" />
<attribute name="charoff" />
<attribute name="span" />
<attribute name="valign" />
<attribute name="width" />
<attribute name="title" />
```

```
            <attribute name="style" />
        </attributes>
    </tag-attributes>
```

The following HTML tags are NOT supported in the "description" field:

XML

```
<tag name="script"/>
<tag name="noscript"/>
<tag name="head"/>
<tag name="select"/>
<tag name="form"/>
<tag name="iframe"/>
<tag name="frame"/>
<tag name="frameset"/>
<tag name="object"/>
<tag name="applet"/>
<tag name="link"/>
```

Instructions on calling APIs

The Seller Center APIs are called through an HTTP request directed to an endpoint. Partners (ISVs) and sellers can use the official Java SDK directly or assemble HTTP requests to call the Seller Center APIs.

This section introduces how to assemble HTTP requests to call the Seller Center APIs.

API calling process

API calls require data for input and return output as the responses. The general steps for calling an API through generating HTTP request are as follows:

1. Populate parameters and values
2. Generate signature
3. Assemble HTTP request
4. Initiate HTTP request
5. Get HTTP response
6. Interpret JSON/XML results

System and API endpoint URLs

For each venture, staging environment and live environment are available for calling APIs. The staging environment is provided to facilitate test phases during the integration project. You must integrate your system with a server correctly. The following table lists the live and staging environment for each venture.

Venture	Environment	URL
Pakistan	Live	https://api.sellercenter.daraz.pk
	Staging	https://api-staging.sellercenter.daraz.pk
Bangladesh	Live	https://api.sellercenter.daraz.com.bd
	Staging	https://api-staging.sellercenter.daraz.com.bd

Sri Lanka	Live	https://api.sellercenter.daraz.lk
	Staging	https://api-staging.sellercenter.daraz.pk
Nepal	Live	https://api.sellercenter.daraz.com.np
	Staging	https://api-staging.sellercenter.daraz.com.np
Myanmar	Live	https://api.sellercenter.shop.com.mm
	Staging	https://api-staging.sellercenter.shop.com.mm

Retrieving the API key

Each API call is performed by a specific Seller Center user (specified by the UserID parameter and signed by a matching API key). Take the following steps to retrieve your API key.

Step 1: Register as a seller on Daraz at:

- For the live environment: https://sellercenter.daraz.xx/seller/register/registration_open (xx represents the country domain name)
- For the staging environment: https://sellercenter-staging.daraz.xx/seller/register/registration_open

Step 2: Log in the Seller Center at:

- For the live environment: https://sellercenter.daraz.xx/(xx represents the country domain name)
- For the staging environment: https://sellercenter-staging.daraz.xx/

Step 3: Click Settings -> User Management.

- Add or remove users under the seller account.
- Specify the role of a user to define API groups that the user can call.
- Ask your vendor manager/key account manager for the API key.
- Each user will have a unique API key when calling an API.

> **Security of the API key**
> You must ensure the security of your API key and do not share the API key with any third party.

Request and responses

While most methods are called via GET, some write methods get additional request data sent via POST. However, sometimes the data that needs to be supplied is more than what can be transported in request parameters. In those cases, additional data is sent to the server using a POST request. The request body must be in XML format. All data (including parameter names and values) must be UTF-8 encoded.

All methods return a response document, which indicates the status of the operation (either Success or Error) and optionally provides results and/or details related to the specified action. The response can be in XML or JSON format.

Common parameters

Common parameters are required in the HTTP request of every API call, listed in the following table:

Field	Type	Description
Action	string	Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	Time when the request is sent, in ISO 8601 format (e.g., Timestamp=2016-04-01T10:00:00+0200). Mandatory.
UserID	string	The ID of the user making the call. The list of authorized users is maintained in the Seller Center web interface under Settings -> Manage Users. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. The signature algorithm is described below. Mandatory.

Business parameters

In addition to the common parameters that must be included in the API call request, the business parameters for the request are also required. Refer to the documentation of each API call for details about the business parameters.

Signature algorithm

Refer to the Signing requests section for detailed information about how to generate signature.

HTTP request sample

Assemble the HTTP request with the following information:

1. API endpoint URLs
2. API call name and return format
3. Public parameters and business parameters
4. Timestamp, user ID, and API version
5. Signature

Taking the *GetCategoryTree* API call as example, the assembled HTTP request call looks like this:

```
https://api.sellercenter.daraz.pk?Action=GetCategoryTree&Format=json&Timestamp=2017-11-25T07%3A00%3A32%2B00%3A00&UserID=xxx&Version=1.0&Signature=xx
```

> **Call limitation**
> You can make at most 10 parallel API calls concurrently.

Signing requests

For the security of data and system, all requests posted to the server must be cryptographically signed.

To ensure that calls to the API cannot be recorded and replayed, one of the parameters getting signed is a timestamp.

More specifically, the signature parameter required on all calls is the HMAC of the request string and your API key with the SHA-256 digest algorithm.

API key

Each user will have a unique API key when calling an API, which is used to sign the request. The API key for a user is created with the creation of the user ID and can be provided to the user who is seeking integration with seller API.

> **Security of the API key**
> You must ensure the security of your API key and do not share the API key with any third party.

Computing the signature parameter

The string to sign is...

- the concatenated result of all request parameters
- ordered by name
- including optional parameters
- and excluding the signature parameter

Names and values must be URL encoded according to RFC 3986 standard, concatenated with the character '='. Each parameter set

(name=value) must be separated with the character '&'.

The following is the reference implementation PHP.

PHP

```php
<?php

// Pay no attention to this statement.
// It's only needed if timezone in php.ini is not set correctly.
date_default_timezone_set("UTC");

// The current time. Needed to create the Timestamp parameter below.
$now = new DateTime();

// The parameters for the GET request. These will get signed.
$parameters = array(
    // The ID of the user making the call.
    'UserID' => 'look@me.com',

    // The API version. Currently must be 1.0
    'Version' => '1.0',

    // The API method to call.
    'Action' => 'GetBrands',

    // The format of the result.
    'Format' => 'XML',

    // The current time in ISO8601 format
    'Timestamp' => $now->format(DateTime::ISO8601)
);

// Sort parameters by name.
ksort($parameters);

// URL encode the parameters.
$encoded = array();
foreach ($parameters as $name => $value) {
    $encoded[] = rawurlencode($name) . '=' . rawurlencode($value);
}

// Concatenate the sorted and URL encoded parameters into a string.
$concatenated = implode('&', $encoded);

// The API key for the user as generated in the Seller Center GUI.
// Must be an API key associated with the UserID parameter.
$api_key = 'b1bdb357ced10fe4e9a69840cdd4f0e9c03d77fe';

// Compute signature and add it to the parameters.
$parameters['Signature'] =
    rawurlencode(hash_hmac('sha256', $concatenated, $api_key, false));
```

If you want to verify the above reference, replace the timestamp with the following value:

PHP

```
'Timestamp' => '2015-07-01T11:11:11+00:00'
```

You should get the following entries in $parameters.

Text

```
Action=GetBrands
Format=XML
Timestamp=2015-07-01T11:11:11+00:00
UserID=look@me.com
Version=1.0
Signature=3ceb8ed91049dfc718b0d2d176fb2ed0e5fd74f76c5971f34cdab484124760
41
```

Then, to make the GET request in PHP, you would write something like this:

PHP

```
<?php

// ...continued from above

// Replace with the URL of your API host.
$url = "https://api.sellercenter.daraz.pk/?";

// Build Query String
$queryString = http_build_query($parameters, '', '&',
PHP_QUERY_RFC3986);

// Open cURL connection
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url."?".$queryString);

// Save response to the variable $data
curl_setopt($ch, CURLOPT_FOLLOWLOCATION,1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
$data = curl_exec($ch);

// Close Curl connection
curl_close($ch);
```

If you are more familiar with other programming languages, here are implementations in...

- Java

- Python
- Visual Basic
- Cold Fusion

Signing in Java SDK

When developing in Java SDK, you do not need to sign every request. The SDK signs them for you. All you need to do is simply initializing the global client.

Java

```
//init global default client with venture URL, your email and API key
DarazClient.init("https://api.sellercenter.daraz.pk/","test@126.com",
"3aac65392ce5ebdfa7eadf4933c9316a4810581e");

//then forget all about signing, just fire your request

//Example
//1. query order by ID
//init request with orderId
GetOrder request = new GetOrder(6642038L);
//fire request and handle result Order
try {
  GetOrderResponse response = request.execute();
  System.out.println(response.getBody());
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}

//2. upload image
//construct request with local file
File image = new File("/Users/yucheng/Desktop/google-search.png");
UploadImage uploadImage = new UploadImage(image);
try {
  ModifyImageResponse response = uploadImage.execute();
  System.out.println("New Url: " +
response.getBody().getImage().getUrl());
  System.out.println("HashCode: " +
response.getBody().getImage().getCode());
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

API call in Java

Java

```
/*
  * Sample Interface for SellerCenter API
```

```
  */
package com.rocket.sellercenter;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.text.SimpleDateFormat;
import java.text.DateFormat;
import java.io.*;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.*;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class SellercenterAPI {
  private static final String ScApiHost =
"http://api.sellercenter.daraz.pk/";
  private static final String HASH_ALGORITHM = "HmacSHA256";
  private static final String CHAR_UTF_8 = "UTF-8";
  private static final String CHAR_ASCII = "ASCII";
  public static void main(String[] args) {
    Map<String, String> params = new HashMap<String, String>();
    params.put("UserID", "example@example.com");
    params.put("Timestamp", getCurrentTimestamp());
    params.put("Version", "1.0");
    params.put("Action", "ProductUpdate");
    final String apiKey = "55f86f79f3b4388507aba8c21a7bfd0d25626551";
    final String XML = "<?xml version=\"1.0\" encoding=\"UTF-8\" ?

><Request><Product><SellerSku>4105382173aaee4</SellerSku><Price>12</Pric
e></Product></Request>";
      final String out = getSellercenterApiResponse(params, apiKey,
XML); // provide XML as an empty string
    when not needed
      System.out.println(out); // print out the XML response
  }

  /**
  * calculates the signature and sends the request
  *
  * @param params Map - request parameters
  * @param apiKey String - user's API Key
  * @param XML String - Request Body
  */
  public static String getSellercenterApiResponse(Map<String, String>
params, String apiKey, String XML) {
    String queryString = "";
    String Output = "";
    HttpURLConnection connection = null;
    URL url = null;
```

```
    Map<String, String> sortedParams = new TreeMap<String,
String>(params);
    queryString = toQueryString(sortedParams);
    final String signature = hmacDigest(queryString, apiKey,
HASH_ALGORITHM);
    queryString = queryString.concat("&Signature=".concat(signature));
    final String request = ScApiHost.concat("?".concat(queryString));
    try {
      url = new URL(request);
      connection = (HttpURLConnection) url.openConnection();
      connection.setDoOutput(true);
      connection.setDoInput(true);
      connection.setInstanceFollowRedirects(false);
      connection.setRequestMethod("POST");
      connection.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
      connection.setRequestProperty("charset", CHAR_UTF_8);
      connection.setUseCaches(false);
      if (!XML.equals("")) {
        connection.setRequestProperty("Content-Length", "" +
Integer.toString(XML.getBytes().length));
        DataOutputStream wr = new
DataOutputStream(connection.getOutputStream());
        wr.writeBytes(XML);
        wr.flush();
        wr.close();
      }
      String line;
      BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
      while ((line = reader.readLine()) != null) {
        Output += line + "\n";
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
    return Output;
  }

  /**
  * generates hash key
  *
  * @param msg
  * @param keyString
  * @param algo
  * @return string
  */
  private static String hmacDigest(String msg, String keyString, String
algo) {
    String digest = null;
```

```
    try {
      SecretKeySpec key = new
SecretKeySpec((keyString).getBytes(CHAR_UTF_8), algo);
      Mac mac = Mac.getInstance(algo);
      mac.init(key);
      final byte[] bytes = mac.doFinal(msg.getBytes(CHAR_ASCII));
      StringBuffer hash = new StringBuffer();
      for (int i = 0; i < bytes.length; i++) {
        String hex = Integer.toHexString(0xFF & bytes[i]);
        if (hex.length() == 1) {
          hash.append('0');
        }
        hash.append(hex);
      }
      digest = hash.toString();
    } catch (UnsupportedEncodingException e) {
      e.printStackTrace();
    } catch (InvalidKeyException e) {
      e.printStackTrace();
    } catch (NoSuchAlgorithmException e) {
      e.printStackTrace();
    }
    return digest;
  }

  /**
  * build querystring out of params map
  *
  * @param data map of params
  * @return string
  * @throws UnsupportedEncodingException
  */
  private static String toQueryString(Map<String, String> data) {
    String queryString = "";
    try{
      StringBuffer params = new StringBuffer();
      for (Map.Entry<String, String> pair : data.entrySet()) {
        params.append(URLEncoder.encode((String) pair.getKey(),
CHAR_UTF_8) + "=");
        params.append(URLEncoder.encode((String) pair.getValue(),
CHAR_UTF_8) + "&");
      }
      if (params.length() > 0) {
        params.deleteCharAt(params.length() - 1);
      }
      queryString = params.toString();
    } catch(UnsupportedEncodingException e){
      e.printStackTrace();
    }
    return queryString;
```

```
}

/**
* returns the current timestamp
* @return current timestamp in ISO 8601 format
*/
private static String getCurrentTimestamp(){
  final TimeZone tz = TimeZone.getTimeZone("UTC");
  final DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mmZ");
  df.setTimeZone(tz);
  final String nowAsISO = df.format(new Date());
```

```
    return nowAsISO;
  }
}
```

API call in Python

Python

```
import urllib
from hashlib import sha256
from hmac import HMAC
from datetime import datetime

parameters = {
  'UserID': 'look@me.com',
  'Version': '1.0',
  'Action': 'FeedList',
  'Format':'XML',
  'Timestamp': datetime.now().isoformat()\n}
api_key = 'b1bdb357ced10fe4e9a69840cdd4f0e9c03d77fe'
concatenated = urllib.urlencode(sorted(parameters.items()))
parameters['Signature'] = HMAC(api_key, concatenated,
sha256).hexdigest()
```

API call in Visual Basic

Visual Basic

```
Imports System
Public Module modmain
 Sub Main()
  ' add your data here:
    Dim userId As String = "" 'login name / your email
    Dim password As String = "" 'your API key/password
    Dim version As String = "1.0"
    Dim action As String = "ProductCreate"
    Dim url As String = ""
    'e.g.: "https://api.sellercenter.daraz.pk/"
    Dim result As String

    ' this is where the magic happens:
    result = generateRequest(url, userId, password, version, action)
    Console.WriteLine (result)
    End Sub
```

```
  Function generateRequest(Url As String, user As String, key as String,
version As String, action As
String) As String
      Dim timeStamp as String =
DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ss-0000")
      ' ATTENTION: parameters must be in alphabetical order
      Dim stringToHash As String = _
      "Action=" + URLEncode(action) + _
      "&Timestamp=" + URLEncode(timeStamp) + _
      "&UserID=" + URLEncode(user) + _
      "&Version=" + URLEncode(version)
      Dim hash As String = HashString(stringToHash, key)
      ' ATTENTION: parameters must be in alphabetical order
      Dim request As String = _
      "Action=" + URLEncode(action) + _
      "&Signature=" + URLEncode(hash) + _
      "&Timestamp=" + URLEncode(timeStamp) + _
      "&UserID=" + URLEncode(user) + _
      "&Version=" + URLEncode(version)
      return url + "?" + request
  End Function

    ' use this function instead of HttpServerUtility.UrlEncode()
    ' because we need uppercase letters
    Function URLEncode(EncodeStr As String) As String
     Dim i As Integer
      Dim erg As String
      erg = EncodeStr
      erg = Replace(erg, "%", Chr(1))
      erg = Replace(erg, "+", Chr(2))
      For i = 0 To 255
        Select Case i
        ' *** Allowed 'regular' characters
          Case 37, 43, 45, 46, 48 To 57, 65 To 90, 95, 97 To 122, 126
          Case 1 ' *** Replace original % erg = Replace(erg, Chr(i),
"%25")
          Case 2 ' *** Replace original + erg = Replace(erg, Chr(i),
"%2B")
          Case 32 erg = Replace(erg, Chr(i), "+")
          Case 3 To 15 erg = Replace(erg, Chr(i), "%0" & Hex(i))
          Case Else
            erg = Replace(erg, Chr(i), "%" & Hex(i))
        End Select
      Next
      return erg
    End Function

    Function HashString(ByVal StringToHash As String, ByVal HachKey As
String) As String
      Dim myEncoder As New System.Text.UTF8Encoding
```

```
      Dim Key() As Byte = myEncoder.GetBytes(HachKey)
      Dim Text() As Byte = myEncoder.GetBytes(StringToHash)
      Dim myHMACSHA256 As New
System.Security.Cryptography.HMACSHA256(Key)
      Dim HashCode As Byte() = myHMACSHA256.ComputeHash(Text)
      Dim hash As String = Replace(BitConverter.ToString(HashCode), "-",
"")
```

```
      Return hash.ToLower
  End Function
End Module
```

API call in Adobe ColdFusion

ColdFusion

```
<!--- this is a sample Adobe ColdFusion script for sending API request
to SellerCenter --->
<!--- hashing function --->
<cffunction name="HMAC_SHA256" returntype="string" access="private"
output="false">
  <cfargument name="Data" type="string" required="true" />
  <cfargument name="Key" type="string" required="true" />
  <cfargument name="Bits" type="numeric" required="false" default="256"
/>
  <cfset var i = 0 />
  <cfset var HexData = "" />
  <cfset var HexKey = "" />
  <cfset var KeyLen = 0 />
  <cfset var KeyI = "" />
  <cfset var KeyO = "" />
  <cfset HexData = BinaryEncode(CharsetDecode(Arguments.data,
"iso-8859-1"), "hex") />
  <cfset HexKey = BinaryEncode(CharsetDecode(Arguments.key,
"iso-8859-1"), "hex") />
  <cfset KeyLen = Len(HexKey)/2 />
  <cfif KeyLen gt 64>
    <cfset HexKey = Hash(CharsetEncode(BinaryDecode(HexKey, "hex"),
"iso-8859-1"), "SHA-256", "iso-8859-1") />
    <cfset KeyLen = Len(HexKey)/2 />
  </cfif>
  <cfloop index="i" from="1" to="#KeyLen#">
    <cfset KeyI = KeyI &
Right("0"&FormatBaseN(BitXor(InputBaseN(Mid(HexKey,2*i-
          1,2),16),InputBaseN("36",16)),16),2) />
    <cfset KeyO = KeyO &
Right("0"&FormatBaseN(BitXor(InputBaseN(Mid(HexKey,2*i-
          1,2),16),InputBaseN("5c",16)),16),2) />
  </cfloop>
  <cfset KeyI = KeyI & RepeatString("36",64-KeyLen) />
  <cfset KeyO = KeyO & RepeatString("5c",64-KeyLen) />
  <cfset HexKey = Hash(CharsetEncode(BinaryDecode(KeyI&HexData, "hex"),
"iso-8859-1"), "SHA-256", "iso-8859-1")
        />
  <cfset HexKey = Hash(CharsetEncode(BinaryDecode(KeyO&HexKey, "hex"),
```

```
"iso-8859-1"), "SHA-256", "iso-8859-

1") />
  <cfreturn Left(HexKey,arguments.Bits/4) />
</cffunction>
<!---/ hashing function --->
<!--- define --->
<cfset send_xml = false><!--- for APIs that need XML request body, like
xxxxProduct APIs --->
  <cfset secret_key="562aeae4090d3a62ef171b6646cc2bdac6417473"/>
  <cfset sc_api_host="http://sellercenter-api.local/"/>
  <!---/ define --->
  <!--- request params --->
  <cfset params = structNew() />
  <cfset params["example@example.com"] = "UserID"/>
  <cfset params["GetShipmentProviders"] = "Action"/>
  <cfset params["2014-07-24T20:06:33+02:00"] = "Timestamp"/>
  <cfset params["1.0"] = "Version"/>
  <cfsavecontent variable="strXML">
    <?xml version="1.0" encoding="UTF-8" ?>
    <Request>
      <Product>
        <SellerSku>4105382173aaee4</SellerSku>
        <Price>12</Price>
      </Product>
    </Request>
  </cfsavecontent>
  <!---/ request params --->
  <!--- generate signature --->
  <cfset strtohash = "" />
  <cfloop list="#ArrayToList(StructSort(params, "text", "asc"))#"
index="key" >
    <cfset strtohash = strtohash & #params[key]# & "=" &
#URLEncodedFormat(key)# & "&" />
  </cfloop>
  <cfset strtohash = #RemoveChars(strtohash, len(strtohash), 1)#/>
  <cfset strtohash = #Replace(strtohash, "%2D", "-", "All")#/>
  <cfset strtohash = #Replace(strtohash, "%2E", ".", "All")#/>
  <cfset signature="#LCase(HMAC_SHA256(strtohash, secret_key))#"/>
  <!---/ generate signature --->
  <!--- issue the request --->
  <cfif send_xml>
    <cfset http_method = "post" />
  <cfelse>
    <cfset http_method = "get" />
  </cfif>
 <cfhttp method="#http_method#" url="#sc_api_host#">
 <cfloop list="#ArrayToList( StructSort(params, "text", "asc") )#"
index="key" >
  <cfhttpparam type="url" name="#params[key]#"
```

```
value="#key#"></cfhttpparam>
 </cfloop>
 <cfhttpparam type="xml" value="#strXML.Trim()#"></cfhttpparam>
 <cfhttpparam type="url" name="Signature"
value="#signature#"></cfhttpparam>
 </cfhttp>
```

```
  <!---/ issue the request --->
  <!--- XML response --->
 <cfoutput>#cfhttp.filecontent#</cfoutput>
```

Requests and responses

While most methods are called via GET, some write methods get additional request data sent via POST.

All methods return a response document, which indicates the status of the operation (either Success or Error) and optionally provides results and/or details related to the specified action.

Additional request data via POST

All API methods are called via HTTP, some via GET and some via POST.

All calls always take the *Action*, *Timestamp*, *UserID*, *Version* and *Signature* parameters. Some additional parameters can also be specified. For example, *GetProducts* takes additional parameters, such as the *Search* parameter.

However, sometimes the data that needs to be supplied is more than what can be transported in request parameters. In those cases, additional data is sent to the server using a POST request. The data must be in XML format (even if the Format parameter is set to 'JSON'). All data (including parameter names and values) must be UTF8-encoded (which should be indicated in the XML preamble) and uploaded as MIME type application/x-www-form-urlencoded.

The following example shows additional data sent to the *ProductUpdate* method:

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request>
  <Product>
    <SellerSku>SKU-AAAA</SellerSku>
    <Price>10.0</Price>
    <SaleStartDate>2015-07-01T11:11:11+0000</SaleStartDate>
    <SaleEndDate>2015-07-01T11:11:11+0000</SaleEndDate>
    <SalePrice>8.0</SalePrice>
  </Product>
  <Product>
    <SellerSku>SKU-BBBB</SellerSku>
    <Price>32.5</Price>
  </Product>
</Request>
```

GET or POST
Whether to use GET or POST depends on the method you are calling and is indicated in the Definition section of each API method in this documentation. Even if no additional data needs to be uploaded for a request, you must use POST if that is the specified HTTP verb for the method.

POST request size limitation
According to the SC standard web server setup, maximum POST request body size is 128MB.

Non-data results

Some methods return long XML documents. For example, *GetProducts* will return a very long XML document, listing each product. The syntax of these responses is explained in detail on the reference page for the method.

A number of methods return no data. In those cases, a response will also be returned, which is called Success Response. Depending on the Format parameter that was passed in, it can be in JSON or XML format. See the following examples.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId>13e55362-3cc4-446b-b3db-c1df0900ae9e</RequestId>
    <RequestAction>PriceFeed</RequestAction>
    <ResponseType></ResponseType>
    <Timestamp>2015-07-01T11:11:11+0000</Timestamp>
  </Head>
  <Body/>
</SuccessResponse>
```

JSON

```
{
  "SuccessResponse": {
    "Head": {
      "RequestId": "13e55362-3cc4-446b-b3db-c1df0900ae9e",
      "RequestAction": "PriceFeed",
      "ResponseType": "",
      "Timestamp": "2015-07-01T11:11:11+0000"
    },
    "Body": ""
  }
}
```

The fields in the <Head> section are always the same, regardless of whether a method returns data in the Body section or not. The meaning of the data in a Success Response is as follows.

Name	Type	Description
RequestId	UUID	The unique identification of this request.
RequestAction	string	The name of the method that was called (i.e., the Action parameter of the request).
ResponseType	string	The response type contained in the Body, or empty if none.
Timestamp	datetime	Time when the request was sent, in ISO 8601 format.
Body	subsection	Additional data as described by the function documentation.

Response and error messages

All API calls will return a response document, which indicates the status of the operation (either Success or Error) and optionally provides results and/or details related to the specified action.

Interpret the response

Some API calls return long XML documents. For example, the *GetProducts* call will return a very long XML document, listing each product. The syntax of these responses is explained in detail on the reference page for the method.

A number of methods return no data. In those cases, a response will also be returned, which is called Success Response. Depending on the Format parameter that was passed in, it can be in JSON or XML format. See the following examples.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId>13e55362-3cc4-446b-b3db-c1df0900ae9e</RequestId>
    <RequestAction>PriceFeed</RequestAction>
    <ResponseType></ResponseType>
    <Timestamp>2015-07-01T11:11:11+0000</Timestamp>
  </Head>
  <Body/>
</SuccessResponse>
```

JSON

```
{
  "SuccessResponse": {
    "Head": {
      "RequestId": "13e55362-3cc4-446b-b3db-c1df0900ae9e",
      "RequestAction": "PriceFeed",
      "ResponseType": "",
      "Timestamp": "2015-07-01T11:11:11+0000"
    },
    "Body": ""
  }
}
```

Similarly, there is an Error Response. It also has a <Body> section, which is often empty, but can be used to provide additional information about the error. Here is an example error message that has a Body:

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<ErrorResponse>
  <Head>
    <RequestAction>Price</RequestAction>
    <ErrorType>Sender</ErrorType>
    <ErrorCode>1000</ErrorCode>
    <ErrorMessage>Format Error Detected</ErrorMessage>
  </Head>
  <Body>
    <ErrorDetail>
      <Field>StandardPrice</Field>
      <Message>Field must contain a positive number with a dot as
decimal
        separator and 2 decimals (e.g. 120.00)
      </Message>
      <Value>10.0x</Value>
      <SellerSku>Example Seller SKU</SellerSku>
    </ErrorDetail>
  </Body>
</ErrorResponse>
```

JSON

```
Response:
{
  "ErrorResponse": {
    "Head": {
      "RequestAction": "ProductUpdate",
      "ErrorType": "Platform",
      "ErrorCode": "1000",
      "ErrorMessage": "Could not save product: 0, A exact match of the
document is being processed"
    },
    "Body": ""
  }
}
```

The meaning of the data in an Error Response is as follows:

Name	Type	Description
RequestAction	string	The API call that triggered the error.
ErrorType	string	The origin of the error (either Sender or Platform).

ErrorCode	integer	The internal error code.
ErrorMessage	string	Error message text.
ErrorDetail	subsection	The error response may contain these subsections in its body, one per error. It contains fields specific to the error. At most 50 ErrorDetails can be provided.

Common error messages

The error messages in the following table are common to all the Seller Center API endpoints. For error messages are specific to an API call, refer to the "Error messages" section of the API call. Error messages are numbered with related message text.

Error code	Message	Explanation
1	E1: Parameter is mandatory	One or more mandatory parameters are not provided in the request. Mandatory parameters for every API call include Action, Timestamp, UserID, and Version.
2	E2: Invalid Version	The provided value for the Version parameter is not correct. The current API version is "1.0".
3	E3: Timestamp has expired	The timestamp should be the correct time, not a time in the past or in the future.
4	E4: Invalid Timestamp format	The timestamp should be the current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00).
5	E5: Invalid Request Format	The input in the request body is not valid. Check the input and try the call again.
6	E6: Unexpected internal error	The error might be caused by invalid input or system error. Check the input and try the call again. If the error persists, submit a support ticket.
7	E7: Login failed. Signature mismatching	The signature is not updated after some changes to the parameters. Regenerate the signature and call the API again.
8	E8: Invalid Action	The API call name might be invalid. Check the spelling of the API call name.
9	E9: Access Denied	The UserID and the signature do not match. Check the correctness of the UserID and the corresponding API key.
1000	E1000: Internal Application Error	The API call failed because of invalid request or system error. Check the request body and try the call again. If the error persists, submit a support ticket.

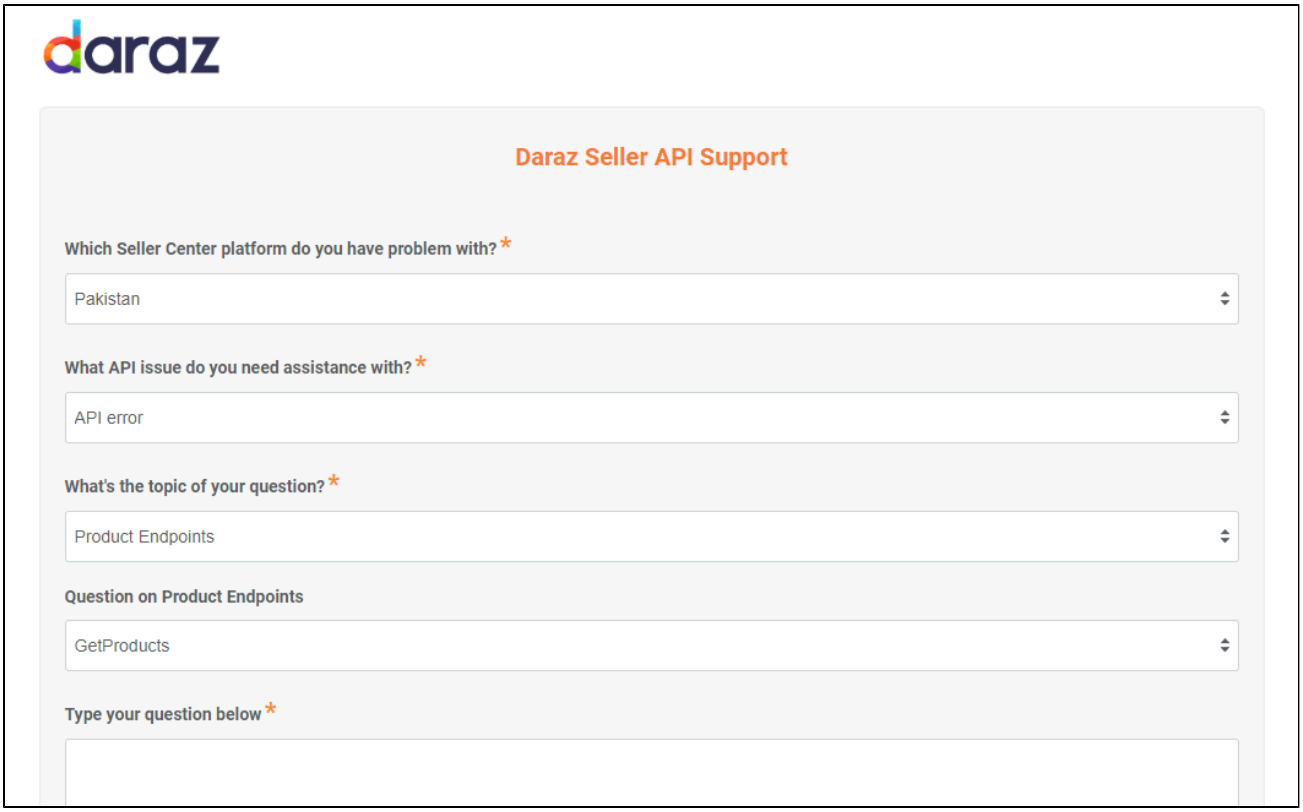
What to do with an error message?

When you get an error, try to investigate and find a fix for the error with the error message and returned information. When your research cannot resolve the the error, you can get technical support by by following the guide of Getting support.

To help the support team locate the specific instance of your request, provide the *RequestID* and *Timestamp* of the failed request.

Getting support

For any questions, contact the support team at API Support and submit the questions.



PRODUCT ENDPOINTS

GetCategoryTree

Definition [GET]

Use this call to retrieve the list of all product categories in the system.

Request URI: https://api.sellercenter.daraz.pk?Action=GetCategoryTree

Parameters

Field	Type	Description
Action	string	*GetCategoryTree* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Code sample

Java

```
//you may find that categoryId of leaf node is useful for creating
product

GetCategoryTree getCategoryTree = new GetCategoryTree();
try {
    GetCategoryTreeResponse response = getCategoryTree.execute();
    List<Category> categories = response.getBody();
    for(Category category: categories) {
        for(Category leaf: category.getLeaves()) {
            System.out.println(leaf.getCategoryId());
        }
    }
} catch (DarazException e) {
    System.out.println(e.getResponseStr());
}
```

Response

The response body is composed of the Categories collection, recursively containing Category resources, each of which is composed of the following fields:

Name	Type	Description
CategoryId	integer	Category ID
Name	string	The name of the category
Children	node	List of all child categories (category resources)
attributeType	string	Type of this attribute. Two possible values: **normal** - means that this attribute belongs to the product **sku** - means that this attribute belongs to the SKU
inputType	string	Type of the attribute value
label	string	Label displayed on the UI for this attribute
name	string	Name of this attribute. It's value is used as the key of this attribute in product creation request.
mandatory	integer	Whether this attribute is mandatory. If its value is 1, this attribute is mandatory. Otherwise not.
options	node	Valid values for this attribute. Useful for an attribute whose inputType is singleSelect or multiSelect.

Result sample

A success result sample is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId/>
    <RequestAction>GetCategoryAttributes</RequestAction>
    <ResponseType>Attributes</ResponseType>
    <Timestamp>2015-07-01T11:11:11+0000</Timestamp>
  </Head>
  <Body>
  [{
    "categoryId":10000012,
    "name":"Books"
  },
  {
    "categoryId":10000009,
    "name":"Phone"
  },
  {
    "categoryId":10000000,
    "name":"autoTest first category"
  },
  {
    "categoryId":10000016,
    "children":[
      {
        "categoryId":10000019,
        "children":[{
          "categoryId":10000020,
          "name":"Games Desktop"
        }],
        "name":"Desktop"
      },
      {
        "categoryId":10000017,
        "name":"Laptop"
      }
    ],
    "name":"Computers"
  },
  {
    "categoryId":10000018,
    "name":"TV"
  }]
  </Body>
</SuccessResponse>
```

Error messages

No specific errors.

GetCategoryAttributes

Definition [GET]

Use this call to get a list of attributes with options for a given category. It will also display attributes for *TaxClass*, with their possible values listed as options.

Request URI: https://api.sellercenter.daraz.pk?Action=GetCategoryAttributes

Parameters

Field	Type	Description
Action	string	*GetCategoryAttributes* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
PrimaryCategory	integer	Identifier of the category for which the caller wants the list of attributes.

Code sample

One exception for category attribute
For all category attributes of type singleSelect or multiSelect, you can find valid values by calling *GetCategoryAttributes*, but this is not true for the 'brand' attribute.
In order to get valid brand list, you need to use the *GetBrands* API.

Java

```
//This API is intend for providing the information listed below:
//1. Attributes for both Product and Sku
//2. Whether or not an attribute is mandatory
//3. Valid values for attribute of type singleSelect or mutiSelect

//you may find this info useful for creating product

GetCategoryAttributes getCategoryAttributes = new
GetCategoryAttributes(2L);
try {
    GetCategoryAttributesResponse response =
getCategoryAttributes.execute();
    System.out.println("Product mandatory attributes:");
    for(Attribute attribute: response.getProductMandatoryAttributes()) {
        System.out.println(String.format("name:[%s],
type:[%s],options:%s",
          attribute.getName(), attribute.getInputType(),
attribute.getOptionValues().toString()));
    }
    System.out.println("Sku mandatory attributes:");
    for(Attribute attribute: response.getSkuMandatoryAttributes()) {
        System.out.println(String.format("name:[%s],
type:[%s],options:%s",
          attribute.getName(), attribute.getInputType(),
attribute.getOptionValues().toString()));
    }
} catch (DarazException e) {
    System.out.println(e.getResponseStr());
}
```

Response

The tags of the result XML have the following types and meaning.

Name	Type	Description
name	string	Identifier of the attribute
label	string	Human-readable display name of the attribute
isMandatory	boolean	Whether the attribute is mandatory. 1 for mandatory, and 0 for optional.
isSaleProp	boolean	Whether the attribute is sale property. 1 for sale property, and 0 for non sale property. Sale property is always mandatory attribute.
Description	string	Attribute description
attributeType	string	Attribute type
inputType	string	Attribute input type (for example, text, rich text, enumInput, multiEnumInput, and option)

options	node	List of all option nodes
name (option)	string	Option name

What are mandatory attributes?
A method to identify mandatory attributes:

Mandatory attribute fields

Eg:
"attributeType": "normal",
"inputType": "singleSelect",
"label": "warranty_type",
"isMandatory": 1

If this is indicated, it would be required.

There will be instances where SKU specific attributes are mandatory.

Result sample

A success result sample is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId/>
    <RequestAction>GetCategoryAttributes</RequestAction>
    <ResponseType>Attributes</ResponseType>
    <Timestamp>2015-07-01T11:11:11+0000</Timestamp>
  </Head>
  <Body>
  [{
    "attributeType":"sku",
    "inputType":"text",
    "isMandatory":1,
    "isSaleProp":0,
    "label":"title",
    "name":"title",
    "options":[]
  },
  {
    "attributeType":"sku",
    "inputType":"richText",
    "isMandatory":1,
    "isSaleProp":0,
    "label":"description",
    "name":"description",
    "options":[]
  },
  {
```

```
    "attributeType":"normal",
    "inputType":"text",
    "isMandatory":1,
    "isSaleProp":0,
    "label":"price",
    "name":"price",
    "options":[]
  },
  {
    "attributeType":"normal",
    "inputType":"option",
    "isMandatory":1,
    "isSaleProp":1,
    "label":"CPU Core",
    "name":"CPU Core",
    "options":[{
        "name":"4 cores"
      },
      {
        "name":"2 cores"
      }]
```

```
  }]
  </Body>
</SuccessResponse>
```

Error messages

Error code	Message
57	E057: No attribute sets linked to that category.

UploadImages

Definition

Use this call to upload multiple image files and accept binary stream with file content. Allowed image formats are JPG and PNG. The maximum size of an image file is 1MB.

Request URI: https://api.sellercenter.daraz.pk/?Action=UploadImages

Parameters

Field	Type	Description
Action	string	*UploadImages* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Request body

Body contains binary image content.

Code sample, response, result sample

The code sample, response, and result sample are similar with those of the *UploadImage* call.

Error messages

Error code	Message
30	E030: Empty Request
300	E300: Upload Image Failed
303	E303: The image is too large
1000	Internal Application Error.

UploadImage

Definition [POST]

Use this call to upload a single image file and accept binary stream with file content. Allowed image formats are JPG and PNG. The maximum size of an image file is 1MB.

Request URI: https://api.sellercenter.daraz.pk?Action=UploadImage

Parameters

Field	Type	Description
Action	string	*UploadImage* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Request body

Body contains binary image content.

Code sample

Text

```
# UploadImage cURL example. to run, update Timestamp and recompute
Signature
#
url = "https://api.sellercenter.daraz.pk/"
post
data-urlencode Action=UploadImage
data-urlencode Timestamp=2016-07-18T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=9ade00fb4b9ab9ed1b8a4d189f1a13e1029edc25dd963235480ec69226fd9f
39
data-urlencode image=@/tmp/image-file.jpg
```

Java

```
//construct request with local file
File image = new File("/Users/yucheng/Desktop/google-search.png");
UploadImage uploadImage = new UploadImage(image);
try {
  ModifyImageResponse response = uploadImage.execute();
  System.out.println("New Url: " +
response.getBody().getImage().getUrl());
  System.out.println("HashCode: " +
response.getBody().getImage().getCode());
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Response

The response body contains an image resource, which is composed of the following fields:

Name	Type	Description
Url	string	The URL address of the uploaded image.
Code	string	The hash code of the image.

Result sample

Success response is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
    <Head>
        <RequestId/>
        <RequestAction>UploadImage</RequestAction>
        <ResponseType>Image</ResponseType>
        <Timestamp>2016-07-07T20:12:14+0700</Timestamp>
    </Head>
    <Body>
        <Image>
            <Url>http://my-live-01.slatic.net/p/orange_yellow.jpg</Url>
            <Code>61bdf049525b7d4c2cf79257ec7c2c56</Code>
        </Image>
    </Body>
</SuccessResponse>
```

JSON


```
{
    "SuccessResponse": {
        "Head": {
            "RequestId": "",
            "RequestAction": "UploadImage",
            "ResponseType": "Image",
            "Timestamp": "2016-08-25T11:26:32+0000"
        },
        "Body": {
            "Image": {
                "Url":
"https//sg.s.alibaba.lzd.co/original/52e91dc864f193c6a096816a0ac1b6d0.jp
g",
                "Code": "52e91dc864f193c6a096816a0ac1b6d0"
            }
        }
    }
}
```


Error messages

Error code	Message
30	E030: Empty Request
300	E300: Upload Image Failed
303	E303: The image is too large
1000	Internal Application Error.

MigrateImage

Definition [POST]

Use this call to migrate a single image from an external site to Daraz site. Allowed image formats are JPG and PNG. The maximum size of an image file is 1MB.

Request URI: https://api.sellercenter.daraz.pk?Action=MigrateImage

Parameters

Field	Type	Description
Action	string	*MigrateImage* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.

Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Code sample

Text

```
# MigrateImage cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
post
data-urlencode Action=MigrateImage
data-urlencode Timestamp=2016-07-18T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=9ade00fb4b9ab9ed1b8a4d189f1a13e1029edc25dd963235480ec69226fd9f
39
```

Java

```
//construct migrate request with url of the image
MigrateImage migrateImage = new
MigrateImage("http://www.google.cn/landing/cnexp/google-search.png");
try {
  ModifyImageResponse response = migrateImage.execute();
  System.out.println("New URL: " +
response.getBody().getImage().getUrl());
  System.out.println("HashCode: " +
response.getBody().getImage().getCode());
} catch (DarazException e) {
  e.printStackTrace();
}
```

Response

The response body contains an image resource, which is composed of the following fields.

Name	Type	Description
Url	string	The URL address of the uploaded image.

Code	string	The hash code of the image.

Result sample

An example of successful response is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId>
    </RequestId>
    <RequestAction>MigrateImage</RequestAction>
    <ResponseType>Image</ResponseType>
    <Timestamp>2016-08-25T11:21:46+0000</Timestamp>
  </Head>
  <Body>
    <Image>

<Url>https//sg.s.alibaba.lzd.co/original/1e8bb2499d38084ffe31f155c68e0d1
f.jpg</Url>
      <Code>1e8bb2499d38084ffe31f155c68e0d1f</Code>
    </Image>
  </Body>
</SuccessResponse>
```

Java


```
{
    "SuccessResponse": {
        "Head": {
            "RequestId": "",
            "RequestAction": "MigrateImage",
            "ResponseType": "Image",
            "Timestamp": "2016-08-25T11:23:29+0000"
        },
        "Body": {
            "Image": {
                "Url":
"https//sg.s.alibaba.lzd.co/original/1e8bb2499d38084ffe31f155c68e0d1f.jp
g",
                "Code": "1e8bb2499d38084ffe31f155c68e0d1f"
            }
        }
    }
}
```


Error messages

Error code	Message
5	E005: Invalid Request Format
6	E006: Unexpected internal error
30	E030: Empty Request
301	E301: Migrate Image Failed
302	E302: Not supported URL.
303	E303: The image is too large
901	E901: The request is too frequent, or the requested functionality is temporarily disabled.
1000	Internal Application Error.

MigrateImages

Definition [POST]

Use this call to migrate multiple images from an external site to Daraz site. Allowed image formats are JPG and PNG. The maximum size of an image file is 1MB. A single call can migrate 8 images at most.

Request URI: https://api.sellercenter.daraz.pk?Action=MigrateImages

Parameters

Field	Type	Description
Action	string	*MigrateImages* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Code sample, response, result sample

The code sample, response, and result sample are similar with those of the *MigrateImage* call.

Error messages

Error code	Message
5	E005: Invalid Request Format
6	E006: Unexpected internal error
30	E030: Empty Request
301	E301: Migrate Image Failed
302	E302: Not supported URL.
303	E303: The image is too large
901	E901: The request is too frequent, or the requested functionality is temporarily disabled.
1000	Internal Application Error.

GetResponse

Definition [POST]

Use this call to get the returned information from the system for the *UploadImages* and *MigrateImages* API.

Request URI: https://api.sellercenter.daraz.pk/?Action=GetResponse

Field	Type	Description
Action	string	*GetResponse* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.

Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Code sample

Text

```
# GetResponse cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
post
data-urlencode Action=GetResponse
data-urlencode Timestamp=2016-07-18T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=9ade00fb4b9ab9ed1b8a4d189f1a13e1029edc25dd963235480ec69226fd9f
39
```

Request body

Enter the request ID from the *UploadImages* or *MigrateImages* API call in the request body.

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request>
    <RequestId>0a1430171490871533268 7334e</RequestId>
</Request>
```

Result format

A success example is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
     <Head>
          <RequestId/>
          <RequestAction>GetResponse</RequestAction>
          <ResponseType>Images</ResponseType>
          <Timestamp>2016-07-07T20:12:14+0700</Timestamp>
     </Head>
     <Body>
     ...
     <Body/>
</SuccessResponse>
```

JSON

```
{
    "SuccessResponse": {
        "Head": {
            "RequestId": "",
            "RequestAction": "GetResponse",
            "ResponseType": "Images",
            "Timestamp": "2016-08-26T06:57:26+0000"
        },
        "Body": {
            "Warnings": []
        }
    }
}
```

Error messages

Error code	Message
5	E005: Invalid Request Format
6	E006: Unexpected internal error

GetBrands

Definition [GET]

Use this call to retrieve all product brands in the system.

Request URI: https://api.sellercenter.daraz.pk?Action=GetBrands

Parameters

Field	Type	Description
Action	string	*GetBrands* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
Limit	integer	The maximum number of brands that can be returned. If you omit this parameter, the default of 100 is used. The Maximum is 1,000.
Offset	integer	Number of brands to skip (i.e., an offset into the result set; together with the Limit parameter, simple result set paging is possible; if you do page through results, note that the list of brands might change during paging).

Code sample

Text

```
# GetBrands cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=GetBrands
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode Limit=100
data-urlencode Offset=0
data-urlencode
Signature=01286525c2fdba58296a61ed1f5e559d0e25972c288d2464b6c25fccc6be72
9a
```

Java

```
//init request with offset(0) and limit(1000)
GetBrands getBrands = new GetBrands(0, 1000);
try {
  GetBrandsResponse response = getBrands.execute();
  for(Brand brand: response.getBody()) {
    System.out.println(brand);
  }
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Response

Field	Type	Description
BrandId	integer	Identifier of this brand as assigned by the Seller Center.
Name	string	The actual name of the brand.
GlobalIdentifier	string	A unique string identifier for the brand across different systems. For example: ADIDAS, NIKE, APPLE.

Result sample

A success result sample is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId/>
    <RequestAction>GetBrands</RequestAction>
    <ResponseType>Brands</ResponseType>
    <Timestamp>2015-07-01T11:11:11+0000</Timestamp>
  </Head>
  <Body>
    <Brands>
      <Brand>
        <BrandId>1</BrandId>
        <Name>Commodore</Name>
        <GlobalIdentifier>commodore</GlobalIdentifier>
      </Brand>
      <Brand>
        <BrandId>2</BrandId>
        <Name>Atari</Name>
        <GlobalIdentifier/>
      </Brand>
    </Brands>
  </Body>
</SuccessResponse>
```

JSON


```
{
  "SuccessResponse": {
    "Head": {
      "RequestAction": "GetBrands",
      "ResponseType": "Brands",
      "Timestamp": "2015-07-01T11:11:11+0000"
    },
    "Body": {
      "Brands": [
          {
            "BrandId": "1",
            "Name": "Commodore",
            "GlobalIdentifier": "commodore"
          },
          {
            "BrandId": "2",
            "Name": "Atari"
          }
        ]
      }
    }
  }
```

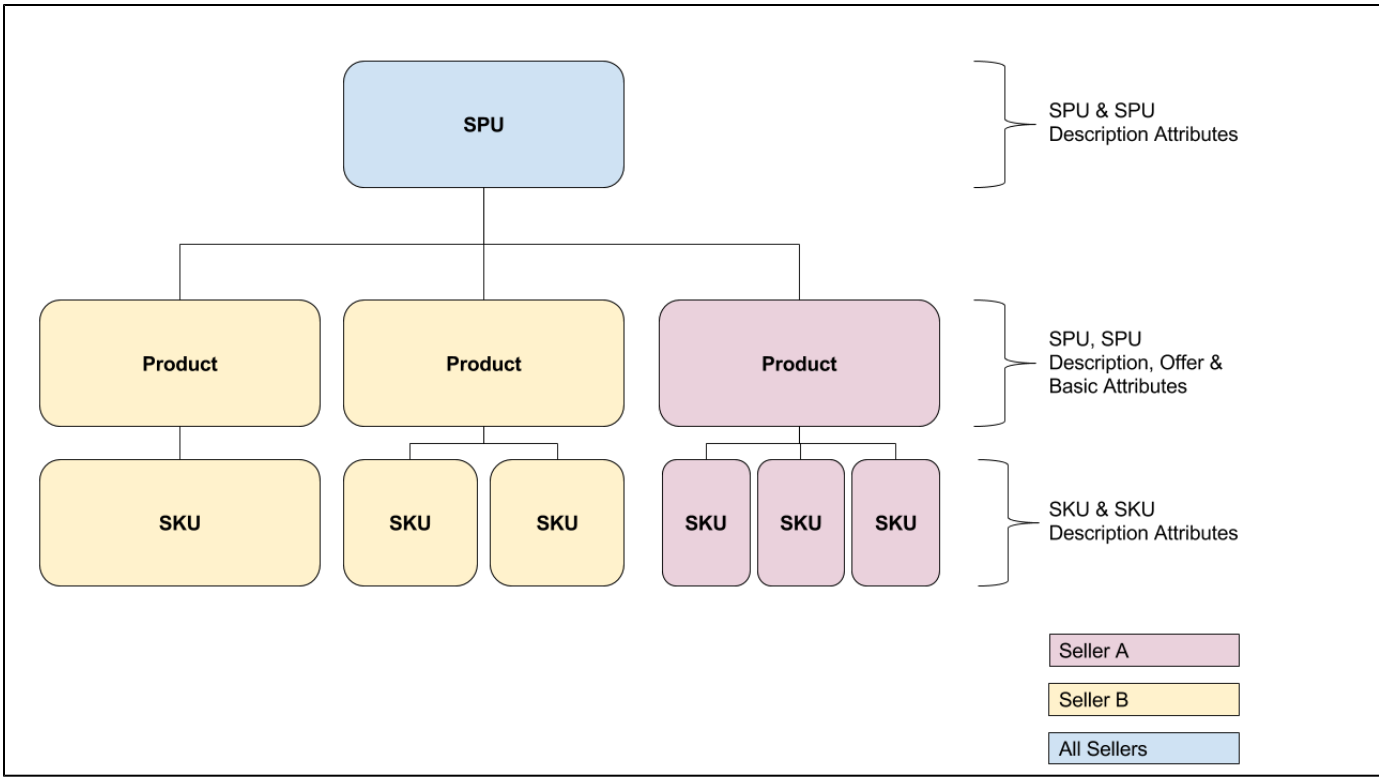

Error messages

No specific errors.

CreateProduct

Product data structure

Before creating a product via API, it is helpful to understand the content data structure.



The highest level of an SKU is SPU. A product inherits attributes from the SPU, and the SKU inherits attributes from the product.

The difference between an SKU and its product is the SKU attributes (which are mandatory) fields when you create your request.

Similarly, product attributes would be mandatory.

Definition

Use this call to create a new product. One item may contain at lest one SKU which has 8 images. This API does not support creating multiple products in one request.

Request URI: https://api.sellercenter.daraz.pk?Action=CreateProduct

Parameters

> Please note that SKU attributes are mandatory for all products created.

Field	Type	Description
Action	string	*CreateProduct* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Code sample

Text

```
# CreateProduct cURL example. to run, update Timestamp and recompute
Signature
#
url = "https://api.sellercenter.daraz.pk/"
post
data-urlencode Action=CreateProduct
data-urlencode Timestamp=2016-07-18T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=9ade00fb4b9ab9ed1b8a4d189f1a13e1029edc25dd963235480ec69226fd9f
39
```

Business parameters

The following tables list the parameters that are required for creating a product.

Name	Type	Description
Product	subsection	The product data node. Mandatory
PrimaryCategory	integer	The ID of the primary category for the product. To get the ID for each of the system's categories, call *GetCategoryTree*. Mandatory. It's optional if 'AssociatedSku' is provided.
SPUId	integer	The ID of the SPU. Optional
AssociatedSku	string	The unique identifier of a product that is already in the system, with which this product should be associated. Optional
Attributes	subsection	All common attributes of products. Mandatory. It's optional if 'AssociatedSku' is provided.
Skus	subsection	An array contains at least one SKU. Mandatory

You can use 'AssociatedSku' if you want to add some SKUs to an existing product. For example, an existing product (Maxi dress) with 3 SKUs (for size S, M, L). SellerSkus are dress-001, dress-002, dress-003.

You can add SKU via the *CreateProduct* API by specifying dress-001, dress-002, or dress-003 in the <AssociatedSku> tag.

The content of the 'Attributes" fields are dynamic. To view all available attributes, call the *GetCategoryAttributes* API. The following table provides some examples.

Name	Type	Description
name	string	Name of the product as shown to the customers. Mandatory. Must be between 2 to 255 characters. If this attribute in SPU you used, it becomes not required.
description	string	Description of the product, as shown to the customers (6 to 25000 characters). Certain HTML tags are supported, but must be escaped as character data (see Guide for Creating Products for the list of supported HTML tags]). Optional
short_description	string	Highlights of the product. Mandatory.
brand	string	Brand name of the product. Mandatory.
model	string	The model name of the product. Optional
warranty	string	The warranty time for the product. Optional

warranty_type	string	The warranty type of the product. Optional
color_family	string	Color family. Optional
...		Other attributes defined in the PrimaryCategory.

An SKU contains the following tags.

Name	Type	Description
SellerSku	string	A unique identifier for the product within the Seller Center instance that is to be added to the system. This identifier is usually freely assigned. Harmonized identifiers, such as UPC or EAN can be set via ProductId. Mandatory
price	decimal	The product price. Not really a Double, but a Decimal. Mandatory
quantity	integer	The current level of inventory for this product. Optional
special_price	decimal	The (hopefully reduced) price for the product while it is on sale. If special_price is specified, either special_from_date or special_to_date must be given; vice versa, if at least one of special_from_date or special_to_date is specified, special_price is mandatory. Not really a Double, but a Decimal.
special_from_date	datetime	Time and date when the product goes on sale. If passed in, special_price becomes mandatory. The value of 'Time' is accepted in intervals of 15 mins. (For ex: 2017-07-15 18:23 will be converted to 2017-07-15 18:15)
special_to_date	datetime	Time and date when the sale of the product ends. If passed in, special_price becomes mandatory. The value of 'Time' is accepted in intervals of 15 mins. (For ex: 2017-07-15 18:23 will be converted to 2017-07-15 18:15)
package_height	string	Package height. Mandatory
package_length	string	Package length. Mandatory
package_width	string	Package width. Mandatory
package_weight	string	Package weight. Mandatory
package_content	string	Package Content. Optional
Images	subsection	Contains most 8 images URL. Optional
Image	string	The image URL. You can set at most 8 images for an SKU. The first image will be used as *MainImage*. Optional. Use the *UploadImages* or *MigrateImages* call to get the image URLs. This will ensure the success rate of creating products.

Request body

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request>
    <Product>
        <PrimaryCategory>6614</PrimaryCategory>
        <SPUId></SPUId>
        <AssociatedSku></AssociatedSku>
        <Attributes>
            <name>api create product test sample</name>
            <short_description>This is a nice
product</short_description>
            <brand>Remark</brand>
            <model>asdf</model>
```

```
            <kid_years>Kids (6-10yrs)</kid_years>
        </Attributes>
        <Skus>
            <Sku>
                <SellerSku>api-create-test-1</SellerSku>
                <color_family>Green</color_family>
                <size>40</size>
                <quantity>1</quantity>
                <price>388.50</price>
                <package_length>11</package_length>
                <package_height>22</package_height>
                <package_weight>33</package_weight>
                <package_width>44</package_width>
                <package_content>this is what's in the
box</package_content>
                <Images>

<Image>http://sg.s.alibaba.lzd.co/original/59046bec4d53e74f8ad38d1939920
5e6.jpg</Image>

<Image>http://sg.s.alibaba.lzd.co/original/179715d3de39a1918b19eec3279dd
482.jpg</Image>
                </Images>
            </Sku>
            <Sku>
                <SellerSku>api-create-test-2</SellerSku>
                <color_family>Green</color_family>
                <size>41</size>
                <quantity>2</quantity>
                <price>520.88</price>
                <package_length>11</package_length>
                <package_height>22</package_height>
                <package_weight>33</package_weight>
                <package_width>44</package_width>
                <package_content>this is what's in the
box</package_content>
                <Images>

<Image>http://sg.s.alibaba.lzd.co/original/59046bec4d53e74f8ad38d1939920
5e6.jpg</Image>

<Image>http://sg.s.alibaba.lzd.co/original/179715d3de39a1918b19eec3279dd
482.jpg</Image>
                </Images>
            </Sku>
```

```
        </Skus>
    </Product>
</Request>
```

Java

```
//CreateProduct requires three piece of information
//1.Category Id --- which can be queried by GetCategoryTree
//2.Product attributes --- whose key set and value set can be queried by
GetCategoryAttributes
//3.one or more Sku(s) attributes --- the set and value set can be
queried by GetCategoryAttributes

//construct attributes of product
Map<String, Object> attributes = new HashMap<String, Object>();
attributes.put("warranty_type", "No Warranty");
attributes.put("package_height", 11.1);
attributes.put("short_description", "yucheng CreateProduct test");
attributes.put("name", "yucheng's first product");
attributes.put("name_ms", "yucheng's first product");
attributes.put("model", "test model");
attributes.put("brand", "Huawei");
attributes.put("display_size_mobile", "4.3");
attributes.put("operating_system_version", "Android 5.1 Lollipop");
attributes.put("operating_system","Android");

//construct SKUs
List<Map<String, Object>> skusList = new ArrayList<>();
Map<String, Object> sku1 = new HashMap<String, Object>();
sku1.put("SellerSku", "yucheng-test-sku-2017020305");
sku1.put("package_height", 11.1);
sku1.put("short_description", "yucheng CreateProduct test");
sku1.put("package_width", 5.9);
sku1.put("tax_class", "default");
sku1.put("package_content", "empty");
sku1.put("package_weight", 11.9);
sku1.put("package_length", 19.3);
sku1.put("color_family", "Blue");
sku1.put("storage_capacity_new", "128G");
skusList.add(sku1);

Map<String, Object> sku2 = new HashMap<String, Object>();
sku2.put("SellerSku", "yucheng-test-sku-2017020306");
sku2.put("package_height", 11.1);
sku2.put("short_description", "yucheng CreateProduct test");
sku2.put("package_width", 5.9);
sku2.put("tax_class", "default");
sku2.put("package_content", "empty");
```

```
sku2.put("package_weight", 11.9);
sku2.put("package_length", 19.3);
sku2.put("color_family", "Blue");
sku2.put("storage_capacity_new", "64GB");
skusList.add(sku2);

//construct request by categoryId, attributes of product, attributes of
sku(s)
CreateProduct createProduct = new CreateProduct(3L, attributes,
skusList);
try {
  ModifyProductResponse response = createProduct.execute();
  System.out.println(String.format("CreateProduct
succeeded?%b",response.getBody()));
```

```
  } catch (DarazException e) {
    System.out.println(e.getResponseStr());
  }
```

PHP

```
//this is file for put daraz api xml format
$tmpFile = 'test.xml';
///SENDING DATA TO DARAZ API
$curl = curl_init();
//TRUE to HTTP PUT a file. The file to PUT must be set with
CURLOPT_INFILE and CURLOPT_INFILESIZE.
curl_setopt( $curl, CURLOPT_PUT, 1 );
//display headers
curl_setopt( $curl, CURLOPT_HEADER, true);
//The name of a file holding one or more certificates to verify the peer
with. This only makes sense when used in combination with
CURLOPT_SSL_VERIFYPEER.
curl_setopt( $curl, CURLOPT_SSL_VERIFYPEER, false);
//  A directory that holds multiple CA certificates. Use this option
alongside CURLOPT_SSL_VERIFYPEER.
curl_setopt( $curl, CURLOPT_SSL_VERIFYHOST, false);
//  TRUE to HTTP PUT a file. The file to PUT must be set with
CURLOPT_INFILE and CURLOPT_INFILESIZE.
curl_setopt( $curl, CURLOPT_INFILESIZE, filesize($tmpFile) );
//The expected size, in bytes, of the file when uploading a file to a
remote site.
curl_setopt( $curl, CURLOPT_INFILE, ($in=fopen($tmpFile, 'r')) );
//A custom request method to use instead of "GET" or "HEAD" when doing a
HTTP request.
curl_setopt( $curl, CURLOPT_CUSTOMREQUEST, 'POST' );
//An array of HTTP header fields to set,
curl_setopt( $curl, CURLOPT_HTTPHEADER, [ 'Content-Type:
application/x-www-form-urlencoded' ] );
//The URL to fetch.
curl_setopt( $curl, CURLOPT_URL, $target );
//TRUE to return the transfer as a string of the return value of
curl_exec() instead of outputting it out directly.
curl_setopt( $curl, CURLOPT_RETURNTRANSFER, 1 );
//executing code for connection
$result = curl_exec($curl);
//closing connection
curl_close($curl);
//closing the open file CURLOPT_INFILE
fclose($in);
//printing data for user
print $result;
```

Result sample

A success example is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
     <Head>
          <RequestId/>
          <RequestAction>CreateProduct</RequestAction>
          <ResponseType>Product</ResponseType>
          <Timestamp>2016-07-06T20:12:14+0700</Timestamp>
     </Head>
     <Body/>
</SuccessResponse>
```

JSON

```
{
    "SuccessResponse": {
        "Head": {
            "RequestId": "",
            "RequestAction": "CreateProduct",
            "ResponseType": "Product",
            "Timestamp": "2016-08-26T06:57:26+0000"
        },
        "Body": {
            "Warnings": []
        }
    }
}
```

A failure example is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<ErrorResponse>
  <Head>
    <RequestAction>CreateProduct</RequestAction>
    <ErrorType>Platform</ErrorType>
    <ErrorCode>500</ErrorCode>
    <ErrorMessage>E500: Create product failed</ErrorMessage>
  </Head>
  <Body>
    <Errors>
      <ErrorDetail>
        <Field>SellerSku</Field>
        <Message>[SELLER_SKU_IS_EXIST]</Message>
      </ErrorDetail>
    </Errors>
  </Body>
</ErrorResponse>
```

JSON

```
{
    "ErrorResponse": {
        "Head": {
            "RequestAction": "CreateProduct",
            "ErrorType": "Platform",
            "ErrorCode": 500,
            "ErrorMessage": "E500: Create product failed"
        },
        "Body": {
            "Errors": [
                {
                    "Field": "SellerSku",
                    "Message": "[SELLER_SKU_IS_EXIST]"
                }
            ]
        }
    }
}
```

Error messages

Error code	Message

1	E001: Parameter %s is mandatory
5	E005: Invalid Request Format
6	E006: Unexpected internal error
30	E030: Empty Request
201	E201: %s Invalid CategoryId
500	E500: Create product failed
901	E901: The request is too frequent, or the requested functionality is temporarily disabled.
1000	Internal Application Error

SetImages

Definition [POST]

Use this call to set the images for an existing product by associating one or more image URLs with it. System supports a maximum of 100 SellerSkus in one request. The first image passed in becomes the default image of the product. There is a hard limit of at most 8 images per SKU. You can also use the *UpdateProduct* call to set images.

Request URI: https://api.sellercenter.daraz.pk?Action=SetImages

Parameters

Field	Type	Description
Action	string	*SetImages* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Code sample

Text

```
# SetImages cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
post
data-urlencode Action=SetImages
data-urlencode Timestamp=2016-07-18T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=9ade00fb4b9ab9ed1b8a4d189f1a13e1029edc25dd963235480ec69226fd9f
39
```

Java

```
//map from seller sku to image list
Map<String, List<String>> sku2Images = new HashMap<>();
sku2Images.put("100024306",
Arrays.asList("http://id-live-03.slatic.net//cms/banners2016/112016/sbd1
2.jpg",

"http://id-live-03.slatic.net//cms/banners2016/112016/sbd4.jpg"));
sku2Images.put("100024305",
Arrays.asList("http://id-live-03.slatic.net//cms/banners2016/112016/sbd-
banner-2.jpg"));

SetImages createProduct = new SetImages(sku2Images);
try {
  ModifyProductResponse response = createProduct.execute();
  System.out.println(String.format("SetImages
succeeded?%b",response.getBody()));
} catch (DarazException e) {
  e.printStackTrace();
  System.out.println(e.getResponseStr());
}
```

Request body

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request>
  <Product>
    <Skus>
      <Sku>
        <SellerSku>Nike Air white</SellerSku>
        <Images>
          <Image>http://static.somecdn.com/img1.jpeg</Image>
          <Image>http://static.somecdn.com/img2.jpeg</Image>
          <Image>http://static.somecdn.com/img3.jpeg</Image>
          <Image>http://static.somecdn.com/img4.jpeg</Image>
          <Image>http://static.somecdn.com/img5.jpeg</Image>
          <Image>http://static.somecdn.com/img6.jpeg</Image>
          <Image>http://static.somecdn.com/img7.jpeg</Image>
          <Image>http://static.somecdn.com/img8.jpeg</Image>
        </Images>
      </Sku>
      <Sku>
        <SellerSku>Nike Air black</SellerSku>
        <Images>
          <Image>http://static.somecdn.com/img11.jpeg</Image>
          <Image>http://static.somecdn.com/img12.jpeg</Image>
          <Image>http://static.somecdn.com/img13.jpeg</Image>
          <Image>http://static.somecdn.com/img14.jpeg</Image>
          <Image>http://static.somecdn.com/img15.jpeg</Image>
          <Image>http://static.somecdn.com/img16.jpeg</Image>
          <Image>http://static.somecdn.com/img17.jpeg</Image>
          <Image>http://static.somecdn.com/img18.jpeg</Image>
        </Images>
      </Sku>
    </Skus>
  </Product>
</Request>
```

If you want to remove some images, remove the <Image> tags with the image URLs. See the following example.

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request>
  <Product>
    <Skus>
      <Sku>
        <SellerSku>Nike Air white</SellerSku>
        <Images>
          <Image>http://static.somecdn.com/img1.jpeg</Image>
          <Image>http://static.somecdn.com/img2.jpeg</Image>
          <Image>http://static.somecdn.com/img3.jpeg</Image>
          <Image>http://static.somecdn.com/img4.jpeg</Image>
        </Images>
      </Sku>
    </Skus>
  </Product>
</Request>
```

Response

Field	Type	Description
Skus	subsection	An array contains at least one SKU. Mandatory
SellerSku	string	A unique identifier for the product within the Seller Center instance that is to be added to the system. This identifier is usually freely assigned. Harmonized identifiers, such as UPC or EAN can be set via ProductId. Mandatory
Images	subsection	Contains at most 8 image URLs. Optional
Image	string	The image URL. Optional

Result sample

A success example is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
     <Head>
          <RequestId/>
          <RequestAction>SetImages</RequestAction>
          <ResponseType>Product</ResponseType>
          <Timestamp>2016-07-07T20:12:14+0700</Timestamp>
     </Head>
     <Body/>
</SuccessResponse>
```

JSON

```
{
    "SuccessResponse": {
        "Head": {
            "RequestId": "",
            "RequestAction": "SetImages",
            "ResponseType": "Product",
            "Timestamp": "2016-08-26T06:57:26+0000"
        },
        "Body": {
            "Warnings": []
        }
    }
}
```

Error messages

Error code	Message
5	E005: Invalid Request Format
6	E006: Unexpected internal error
30	E030: Empty Request
200	E200: Empty SellerSku
203	E203: Too many images in one SKU
204	E204: Too many SKU in one request
504	E504: Set product Image failed
1000	Internal Application Error

GetProducts

Definition [GET]

Use this call to get all or a range of products.

Request URI: https://api.sellercenter.daraz.pk?Action=GetProducts

Parameters

Field	Type	Description
Action	string	*GetProducts* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
CreatedAfter	datetime	Limits the returned products to those created after or on the specified date, given in ISO 8601 date format. Optional
CreatedBefore	datetime	Limits the returned products to those created before or on the specified date, given in ISO 8601 date format. Optional
UpdatedAfter	datetime	Limits the returned products to those updated after or on the specified date, given in ISO 8601 date format. Optional
UpdatedBefore	datetime	Limits the returned product list to those updated before or on a specified date, given in ISO 8601 date format. Optional
Search	string	Returns the products with the search string contained in the product name and/or Seller SKU.
Filter	string	Returns the products with the status matching this parameter. Possible values are all, live, inactive, deleted, image-missing, pending, rejected, sold-out. Mandatory
Limit	integer	The maximum number of products that can be returned. If no value is specified, use 20 as default. The maximum number of products returned by a single call is 500 (this is the default hard limit, which can be changed per instance). If you need to retrieve more products than that, you have to page through the result set using the Offset parameter.
Options	integer	This value can be used to get more stock information. e.g., Options=1 means contain ReservedStock, RtsStock, PendingStock, RealTimeStock, FulfillmentBySellable.
Offset	integer	Number of products to skip (i.e., an offset into the result set; together with the Limit parameter, simple result set paging is possible; if you do page through results, note that the list of products might change during paging).
SkuSellerList	array of strings	Only products that have the Seller SKU in this list will be returned. Input should be a JSON array. For example, ["Apple 6S Gold", "Apple 6S Black"]. It only matches the whole words.

Code sample

Text

```
# GetProducts cUrl example
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=GetProducts
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=9ade00fb4b9ab9ed1b8a4d189f1a13e1029edc25dd963235480ec69226fd9f
39
```

Java

```
//init global client with API endpoints, userEmail, apikey
DarazClient.init("https://api.sellercenter.daraz.pk/",
                 "1852752325@qq.com",
"kGmfpknoWCyVn78k0O2X4L6t01xhrHGwTafLPIMqt8XogwC69au7xLbZ");

//construct request
//query without filter condition
GetProducts request = new GetProducts();
/*
query with one or more filter conditon(s)
ProductsFilter filter = new ProductsFilter();
filter.createdBefore(new Date(117, 0, 26, 0, 0));
filter.search("ASC Test");
GetProducts request = new GetProducts(filter);
*/
try {
  GetProductsResponse response = request.execute();
  List<Product> products = response.getBody();
} catch (DarazException e) {
  e.printStackTrace();
}
```

Response

The returned results and their type and description are as follows.

Field	Type	Description
Products	subsection	An array contains at least one Product. Mandatory
PrimaryCategory	integer	The ID of the primary category for his product. To get the ID for all categories, call GetCategoryTree. Optional

Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
SPUId	integer	The ID of SPU. Optional
Attributes	subsection	Contains several product attributes. Mandatory
Skus	subsection	An array contains at least one SKU. Mandatory
TotalProducts	integer	The number of total products, it's product level.

The following table lists some example fields in 'Attributes'.

Field	Type	Description
name	string	The name of the product as shown to the customers. Mandatory. Must be between 2 to 255 characters.
short_description	string	The description of the product, as shown to the customers (6 to 25000 characters). Embedding certain HTML tags is allowed, but must be escaped as character data (see below). Optional
brand	string	The brand name of the product. Optional
model	string	The model name of the product. Optional
warranty	string	The warranty time for the product. Optional
warranty_type	string	The warranty type of the product. Optional
color_family	string	Color family. Optional
...		Other attributes defined in the PrimaryCategory.

The following table lists some example fields in 'Sku'. You can also get all the attributes by call *GetCategoryAttributes*.

Name	Type	Description
SellerSku	string	A unique identifier for the product within the Seller Center instance that is to be added to the system. This identifier is usually freely assigned. Harmonized identifiers, such as UPC or EAN can be set via ProductId. Mandatory
ShopSku	string	The ID of SKU in Shop when product is synchronized to shop. Optional
Status	string	One of the following values: 'active', 'inactive' or 'deleted'. Optional. The default value is 'active'.
Url	string	The product URL on the web site. Optional
Available	integer	Available stock. Mandatory
price	decimal	The product price. Not really a Double, but a Decimal. Optional
quantity	integer	The current level of inventory for this product. Optional
ReservedStock	integer	The stock reserved for pending orders in Seller Center. Optional
RealTimeStock	integer	The stock reserved for pending orders in the Shop not imported in Seller Center yet. Optional
FulfillmentBySellable	integer	The sellable stock in the Venture warehouse. Optional
RtsStock	integer	The stock number of order products in Ready-To-Ship. Optional
PendingStock	integer	The stock number of order products in Pending. Optional
special_price	decimal	The (hopefully reduced) price for the product while it is on sale. If SalePrice is specified, either SaleStartDate or SaleEndDate must be given; Vice versa, if at least one of SaleStartDate or SaleEndDate is specified, SalePrice is mandatory. Not really a Double, but a Decimal.

special_from_date	date	Date when special price will become applicable **Note**: To be used only if date is available.
special_from_time	datetime	Date and time when special price will become applicable. The date given under this tag should be the same as that in 'special_from_date' **Note**: To be used if both date and time are available.
special_to_date	date	Date till when special price is applicable. **Note**: To be used only if date is available.
special_to_time	datetime	Date and time till when special price is applicable. The date given under this tag should be same as in 'special_to_date' **Note**: To be used if both date and time are available.
package_length	string	Package length. Optional
package_height	string	Package height. Optional
package_width	string	Package width. Optional
package_weight	string	Package weight. Optional
package_content	string	Package Content. Optional
Images	subsection	Contains at most 8 images URL. Optional
Image	string	The image URL. Optional

You can set at most 8 images for one SKU. The first image will be used as MainImage.

Result sample

A success example is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId>
    </RequestId>
    <RequestAction>GetProducts</RequestAction>
    <ResponseType>Product</ResponseType>
    <Timestamp>2016-08-25T10:07:00+0000</Timestamp>
  </Head>
  <Body>
    <Products>
      <Product>
        <PrimaryCategory>3</PrimaryCategory>
        <SPUId/>
        <Attributes>
          <name>(Telco Set) Apple iPhone 6s Plus 16GB (Space
Gray)</name>
          <description/>
          <short_description><div class="prod_content"> <div
class="prod_details"> <ul class="prd-attributesList ui-listBulleted"><li
class=""><b><span>5.5" Retina HD Display with 3D
```

```
Touch&amp;nbsp;</span></b></li><li class=""><b><span>Fingerprint-resistant
oleophobic coating&amp;nbsp;</span></b></li><li class=""><b><span>A9 chip
with 64-bit architecture&amp;nbsp;</span></b></li><li
class=""><b><span>Ultrafast 4G LTE Advanced
wireless&amp;nbsp;</span></b></li><li class=""><b><span>New 12-megapixel
iSight camera&amp;nbsp;</span></b></li><li class=""><b><span>4k video
recording&amp;nbsp;</span></b></li><li class=""><b><span>iOS 9 with Touch ID
and Apple Pay</span></b></li></ul> </div> </div></short_description>
          <video/>
          <brand>Bean Rester</brand>
          <model>iPhone 6s Plus 16GB (Space Gray)</model>
          <phone_features/>
          <pixel_ppi/>
          <processor_type/>
          <display_size_mobile/>
          <screen_type/>
          <video_resolution/>
          <color_family/>
          <type_of_battery/>
          <operating_system_version/>
          <network_connections/>
          <ram_memory/>
          <camera_back/>
          <camera_front/>
          <condition>New</condition>
          <sim_slots/>
          <attribute test 4/>
          <ceshi/>
          <delivery_option_economy/>
          <warranty_type>Local (Singapore) manufacturer
warranty</warranty_type>
          <warranty>11 Months</warranty>
          <delivery_option_standard/>
          <delivery_option_express/>
        </Attributes>
        <Skus>
          <Sku>
            <SellerSku>(Telco Set) Apple iPhone 6s Plus 16GB Space
Gray</SellerSku>
            <ShopSku>BE494ELAA4WUACANID-9625124</ShopSku>

<Url>http://daraz.pk/(Telco-Set)-Apple-iPhone-6s-Plus-16GB-(Space-Gray)-
8250708.html</Url>
            <Status>inactive</Status>
            <Available>5</Available>
            <special_to_date>2018-05-01</special_to_date>
            <price>1288.0</price>
            <special_from_date>2016-05-13</special_from_date>
            <quantity>5</quantity>
            <special_price>1173.85</special_price>
```

```
            <operating_system>iOS</operating_system>
            <Images>
              <Image/>

<Image>http://sg-live-01.slatic.net//p/telco-set-apple-iphone-6s-plus-16
gb-space-gray-3191-1425247-d10a8dd02929aacc84b03be2317a7175-catalog.jpg<
/Image>
              <Image/>
              <Image/>
              <Image/>
              <Image/>
              <Image/>
              <Image/>
            </Images>
          </Sku>
        </Skus>
      </Product>
```

```
    </Products>
  </Body>
</SuccessResponse>
```

JSON


```
{
    "SuccessResponse": {
        "Head": {
            "RequestId": "",
            "RequestAction": "GetProducts",
            "ResponseType": "Product",
            "Timestamp": "2016-08-25T11:30:27+0000"
        },
        "Body": {
            "Products": [
                {
                    "PrimaryCategory": 3,
                    "Attributes": {
                        "name": "(Telco Set) Apple iPhone 6s Plus 16GB
(Space Gray)",
                        "short_description": "<div
class=\"prod_content\">\n            <div class=\"prod_details\">\n
<ul class=\"prd-attributesList ui-listBulleted\"><li
class=\"\"><b><span>5.5\" Retina HD Display with 3D
Touch </span></b></li><li class=\"\"><b><span>Fingerprint-resistant
oleophobic coating </span></b></li><li class=\"\"><b><span>A9 chip with
64-bit architecture </span></b></li><li class=\"\"><b><span>Ultrafast 4G
LTE Advanced wireless </span></b></li><li class=\"\"><b><span>New
12-megapixel iSight camera </span></b></li><li class=\"\"><b><span>4k
video recording </span></b></li><li class=\"\"><b><span>iOS 9 with Touch
ID and Apple Pay</span></b></li></ul>            </div>\n
</div>",
                        "brand": "Bean Rester",
                        "model": "iPhone 6s Plus 16GB (Space Gray)",
                        "condition": "New",
                        "warranty_type": "Local (Singapore) manufacturer
warranty",
                        "warranty": "11 Months"
                    },
                    "Skus": [
                        {
                            "ShopSku": "BE494ELAA4WUACANID-9625124",
                            "Status": "inactive",
                            "Url":
"http://alice.sg.alibaba.lzd.co/(Telco-Set)-Apple-iPhone-6s-Plus-16GB-(S
pace-Gray)-8250708.html",
                            "Available": 5,
```

```
                            "special_to_date": "2018-05-01",
                            "price": "1288.0",
                            "special_from_date": "2016-05-13",
                            "quantity": "5",
                            "SellerSku": "(Telco Set) Apple iPhone 6s
Plus 16GB Space Gray",
                            "special_price": "1173.85",
                            "operating_system": "iOS",
                            "Images": [
                                "",

"http://sg-live-01.slatic.net//p/telco-set-apple-iphone-6s-plus-16gb-spa
ce-gray-3191-1425247-d10a8dd02929aacc84b03be2317a7175-catalog.jpg",
                                "",
                                "",
                                "",
                                "",
                                "",
                                ""
                            ]
                        }
                    ]
                },
                {
                    "PrimaryCategory": 3,
                    "ShopSku": "BE494ELAA4WUA2ANID-9625114",
                    "Status": "inactive",
                    "Url":
"http://alice.sg.alibaba.lzd.co/(Telco-Set)-Apple-iPhone-6s-Plus-64GB-(G
old)---8250698.html",
                    "Available": 5,
                    "Attributes": {
                        "name": "(Telco Set) Apple iPhone 6s Plus 64GB
(Gold)  ",
                        "short_description": "<div
class=\"prod_content\">\n            <div class=\"prod_details\">\n
<ul class=\"prd-attributesList ui-listBulleted\"><li
class=\"\"><b><span>5.5\" Retina HD Display with 3D
Touch </span></b></li><li class=\"\"><b><span>Fingerprint-resistant
oleophobic coating </span></b></li><li class=\"\"><b><span>A9 chip with
64-bit architecture </span></b></li><li class=\"\"><b><span>Ultrafast 4G
LTE Advanced wireless </span></b></li><li class=\"\"><b><span>New
12-megapixel iSight camera </span></b></li><li class=\"\"><b><span>4k
video recording </span></b></li><li class=\"\"><b><span>iOS 9 with Touch
ID and Apple Pay</span></b></li></ul>            </div>\n
</div>",
                        "brand": "Bean Rester",
                        "model": "iPhone 6s Plus 64GB (Gold)  ",
                        "condition": "New",
                        "warranty_type": "Local (Singapore) manufacturer
```

```
warranty",
                        "warranty": "11 Months"
                    },
                    "Skus": [
                        {
                            "special_to_date": "2018-05-01",
                            "price": "1558.0",
                            "special_from_date": "2016-05-13",
                            "quantity": "5",
                            "SellerSku": "(Telco Set) Apple iPhone 6s
Plus 64GB Gold",
                            "special_price": "1287.95",
                            "operating_system": "iOS",
                            "Images": [
                                "",

"http://sg-live-01.slatic.net//p/telco-set-apple-iphone-6s-plus-64gb-gol
d-3654-5425247-cc1f1de5663d894bf8e19b95666f832d-catalog.jpg",
                                "",
                                "",
                                "",
                                "",
                                "",
                                "",
                                ""
                            ]
                        }
                    ]
                }
            ]
```

```
            }
        }
    }
```

Error messages

Error code	Message
5	E005: Invalid Request Format
6	E006: Unexpected internal error
14	E014: "%s" Invalid Offset
17	E017: "%s" Invalid Date Format
19	E019: "%s" Invalid Limit
36	E036: Invalid status filter
70	E070: You have corrupt data in your sku seller list.
506	E506: Get product failed
901	E901: The request is too frequent, or the requested functionality is temporarily disabled.

Note
The list of products may change while paging through it using multiple calls with a changing Offset parameter.

UpdateProduct

Definition [POST]

Use this call to update attributes or SKUs of an existing product. Note that one request can update only 1 product.

Request URI: https://api.sellercenter.daraz.pk?Action=UpdateProduct

Note: The following fields cannot be updated with the *UpdateProduct* call:

1. Basic product rules, such as field changes cross categories.
2. Product system fields, like "gmt_modified" and "gmt_create".
3. Special fields on product web page, like "keywords".

Parameters

Field	Type	Description
Action	string	*UpdateProduct* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.

Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Code sample

Text

```
# UpdateProduct cURL example. to run, update Timestamp and recompute
Signature
#
url = "https://api.sellercenter.daraz.pk/"
post
data-urlencode Action=UpdateProduct
data-urlencode Timestamp=2016-07-18T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=9ade00fb4b9ab9ed1b8a4d189f1a13e1029edc25dd963235480ec69226fd9f
39
```

Java

```
//construct attributes of product
Map<String, Object> attributes = new HashMap<String, Object>();
attributes.put("name", "yucheng's first product");
attributes.put("package_height", 11.3);

//construct SKUs
List<Map<String, Object>> skusList = new ArrayList<>();
Map<String, Object> sku1 = new HashMap<String, Object>();
sku1.put("SellerSku", "yucheng-test-sku-2017020305");
sku1.put("package_height", 11.3);
sku1.put("storage_capacity_new", "128G");
skusList.add(sku1);

Map<String, Object> sku2 = new HashMap<String, Object>();
sku2.put("SellerSku", "yucheng-test-sku-2017020306");
sku2.put("package_height", 11.2);
skusList.add(sku2);

UpdateProduct request = new UpdateProduct(attributes, skusList);
try {
  ModifyProductResponse response = request.execute();
  System.out.println("UpdateProduct succeed?" + response.getBody());
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Request body

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<Request>
    <Product>
        <Attributes>
            <name>api update product sample</name>
            <short_description>This is an amazing
product</short_description>
        </Attributes>
        <Skus>
            <Sku>
                <SellerSku>api-create-test-1</SellerSku>
                <quantity>88</quantity>
                <price>350</price>
                <package_length>12</package_length>
                <package_height>23</package_height>
                <package_weight>34</package_weight>
                <package_width>45</package_width>
                <Images></Images>
            </Sku>
            <Sku>
                <SellerSku>api-create-test-2</SellerSku>
                <quantity>44</quantity>
                <price>488.88</price>
                <package_length>10</package_length>
                <package_height>21</package_height>
                <package_weight>32</package_weight>
                <package_width>43</package_width>
                <package_content>this is what's in the box,
update</package_content>
                <Images>

<Image>http://sg.s.alibaba.lzd.co/original/59046bec4d53e74f8ad38d19399205e6.jpg</Image>

<Image>http://sg.s.alibaba.lzd.co/original/179715d3de39a1918b19eec3279dd482.jpg</Image>

<Image>http://sg.s.alibaba.lzd.co/original/e2ae2b41afaf310b51bc5764c17306cd.jpg</Image>
                </Images>
            </Sku>
        </Skus>
    </Product>
</Request>
```

Response

Name	Type	Description
Product	subsection	The product data node. Mandatory
PrimaryCategory	integer	The ID of the primary category for this product. To get the ID of all categories, call GetCategoryTree. Optional
SPUId	integer	The ID of the SPU. Optional
Attributes	subsection	All common attributes of products. Mandatory. It's optional if 'AssociatedSku' is provided.
Skus	subsection	An array contains at least one SKU. Mandatory

Some example fields in 'Attributes' are as follows. These attributes are dynamic. For the list of all attributes, call API *GetCategoryAttributes*.

Name	Type	Description
name	string	Name of the product as shown to the customers. Optional
description	string	Description of the product, as shown to the customers (6 to 25000 characters). Embedding certain HTML tags is allowed, but must be escaped as character data (see below). Optional
short_description	string	Highlights of the product. Mandatory.
brand	string	Brand name of the product. Mandatory.
model	string	The model name of the product. Optional
warranty	string	The warranty time for the product. Optional
warranty_type	string	The warranty type of the product. Optional
color_family	string	Color family. Optional
...		Other attributes defined in the PrimaryCategory. Optional

An SKU contains the following tags.

Name	Type	Description
SellerSku	string	A unique identifier for the product within the Seller Center instance that is to be added to the system. This identifier is usually freely assigned. Harmonized identifiers, such as UPC or EAN can be set via ProductId. Mandatory
active	boolean	One of the following values: 'true' or 'false'. Optional
price	decimal	The product price. Not really a Double, but a Decimal. Mandatory
quantity	integer	The current level of inventory for this product. Optional
special_price	decimal	The (hopefully reduced) price for the product while it is on sale. If special_price is specified, either special_from_date or special_to_date must be given; vice versa, if at least one of special_from_date or special_to_date is specified, special_price is mandatory. Not really a Double, but a Decimal. Optional
special_from_date	datetime	Time and date when the product goes on sale. If passed in, special_price becomes mandatory. The value of 'Time' is accepted in intervals of 15 mins. (For ex: 2017-07-15 18:23 will be converted to 2017-07-15 18:15)
special_to_date	datetime	Time and date when the sale of the product ends. If passed in, special_price becomes mandatory. The value of 'Time' is accepted in intervals of 15 mins. (For ex: 2017-07-15 18:23 will be converted to 2017-07-15 18:15)
package_height	string	Package height. Mandatory
package_length	string	Package length. Mandatory

package_width	string	Package width. Mandatory
package_weight	string	Package weight. Mandatory
package_content	string	Package Content. Optional
Images	subsection	Contains most 8 images URL. Optional
Image	string	The image URL. You can set at most 8 images for an SKU. The first image will be used as *MainImage*. Optional. Use the *UploadImages* or *MigrateImages* call to get the image URLs. This will ensure the success rate of creating products.

Result sample

A success example is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
     <Head>
          <RequestId/>
          <RequestAction>UpdateProduct</RequestAction>
          <ResponseType>Product</ResponseType>
          <Timestamp>2016-07-07T20:12:14+0700</Timestamp>
     </Head>
     <Body/>
</SuccessResponse>
```

JSON

```
{
    "SuccessResponse": {
        "Head": {
            "RequestId": "",
            "RequestAction": "UpdateProduct",
            "ResponseType": "Product",
            "Timestamp": "2016-07-07T20:12:14+0700"
        },
        "Body": {
            "Warnings": []
        }
    }
}
```

A failure example is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<ErrorResponse>
     <Head>
          <RequestAction>UpdateProduct</RequestAction>
          <ErrorType>Platform</ErrorType>
          <ErrorCode>1000</ErrorCode>
          <ErrorMessage>Format Error Detected</ErrorMessage>
     </Head>
     <Body/>
</ErrorResponse>
```

Error messages

Error code	Message
1	E001: Parameter %s is mandatory
5	E005: Invalid Request Format
6	E006: Unexpected internal error
30	E030: Empty Request
201	E201: %s Invalid CategoryId
202	E202: %s Invalid SPUId
501	E501: Update product failed
901	E901: The request is too frequent, or the requested functionality is temporarily disabled.
1000	Internal Application Error

UpdatePriceQuantity

Definition [GET]

Use this call to update the price and quantity of one or more existing products. The maximum number of products that can be updated is 50, but 20 is recommended.

Request URI: https://api.sellercenter.daraz.pk?Action=UpdatePriceQuantity

> **Note**
> A price update of greater than 80% or more of the previous price will result in the product going back to QC.

Parameters

Field	Type	Description
Action	string	*UpdatePriceQuantity* Name of the API that is to be called. Mandatory.

Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Code sample

Text

```
# UpdatePriceQuantity cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
post
data-urlencode Action=UpdatePriceQuantity
data-urlencode Timestamp=2016-07-18T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=9ade00fb4b9ab9ed1b8a4d189f1a13e1029edc25dd963235480ec69226fd9f
39
```

Request body

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<Request>
  <Product>
    <Skus>
      <Sku>
        <SellerSku>Apple-SG-Glod-64G</SellerSku>
        <Quantity>10</Quantity>
        <Price/>
        <SalePrice/>
        <SaleStartDate/>
        <SaleEndDate/>
      </Sku>
      <Sku>
        <SellerSku>Apple-SG-Black-64G</SellerSku>
        <Quantity/>
        <Price>1299.00</Price>
        <SalePrice>1100.00</SalePrice>
        <SaleStartDate/>
        <SaleEndDate/>
      </Sku>
      <Sku>
        <SellerSku>Apple-SG-RoseGold-128G</SellerSku>
        <Quantity>10</Quantity>
        <Price>1888.00</Price>
        <SalePrice>1600.00</SalePrice>
        <SaleStartDate>2016-08-08</SaleStartDate>
        <SaleEndDate>2016-08-31</SaleEndDate>
      </Sku>
    </Skus>
  </Product>
</Request>
```

Java

```
//not released yet
//This api is used for update price and/or quantity for one or more
sku(s)
SkuPriceQuantityInfo updateInfo = new SkuPriceQuantityInfo();
//update price and quantity at one shot
updateInfo.add(new SkuPriceQuantity("yucheng-test-sku-2017020301", new
BigDecimal("11.3"), 10));
//update quantity only
updateInfo.add(new SkuPriceQuantity("yucheng-test-sku-2017020302", 20));
//update price only
updateInfo.add(new SkuPriceQuantity("yucheng-test-sku-2017020303", new
BigDecimal("22.11")));

UpdatePriceQuantity updatePriceQuantity = new
UpdatePriceQuantity(updateInfo);
try {
  ModifyProductResponse response = updatePriceQuantity.execute();
  System.out.println(String.format("update request
succeeded?%b",response.getBody()));
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Business parameters

The request data is composed of the Skus collection, recursively containing Sku resources, each of them contain the following fields.

Name	Type	Description
SellerSku	string	A unique identifier for the product within the Seller Center instance that is to be added to the system. This identifier is usually freely assigned. Harmonized identifiers, such as UPC or EAN can be set via ProductId. Mandatory
Quantity	integer	The current level of inventory for this product. Optional
Price	decimal	The product price. Not really a Double, but a Decimal. Optional
SalePrice	decimal	The (hopefully reduced) price for the product while it is on sale. If SalePrice is specified, either SaleStartDate or SaleEndDate must be given; vice versa, if at least one of SaleStartDate or SaleEndDate is specified, SalePrice is mandatory. Not really a Double, but a Decimal. Optional
SaleStartDate	datetime	Time and date when the product goes on sale. If passed in, SalePrice becomes mandatory. The value of 'Time' is accepted in intervals of 15 mins. (For ex: 2017-07-15 18:23 will be converted to 2017-07-15 18:15)
SaleEndDate	datetime	Time and date when the product sale ends. If passed in, SalePrice becomes mandatory. The value of 'Time' is accepted in intervals of 15 mins. (For ex: 2017-07-15 18:23 will be converted to 2017-07-15 18:15)

Result sample

A success response is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId/>
    <RequestAction>UpdatePriceQuantity</RequestAction>
    <ResponseType>Product</ResponseType>
    <Timestamp>2016-07-07T20:12:14+0700</Timestamp>
  </Head>
  <Body/>
</SuccessResponse>
```

JSON

```
{
    "SuccessResponse": {
        "Head": {
            "RequestId": "",
            "RequestAction": "UpdatePriceQuantity",
            "ResponseType": "Product",
            "Timestamp": "2016-07-07T20:12:14+0700"
        },
        "Body": {
            "Warnings": []
        }
    }
}
```

Error messages

If you receive internal errors while trying to update the price and quantity, it may be because your product data has not been updated properly and it has some mandatory fields not updated. Try the *UpdateProduct* call and then the *UpdatePriceQuantity* call.

Error code	Message
1000	Could not save product: %s
5	E005: Invalid Request Format
6	E006: Unexpected internal error
30	E030: Empty Request
204	E204: Too many SKU in one request
501	E501: Update product failed
901	E901: The request is too frequent, or the requested functionality is temporarily disabled.

1000	Internal Application Error

SALES ORDER ENDPOINTS

GetOrder

Definition [GET]

Use this call to get the order details for a single order. It is different from *GetOrders*, which gets the customer details of multiple orders.

Request URI: https://api.sellercenter.daraz.pk?Action=GetOrder

Parameters

Field	Type	Description
Action	string	*GetOrder* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
OrderId	long	The identifier that was assigned to the order by the Seller Center.

Code sample

Text

```
# GetOrder cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=GetOrder
data-urlencode OrderId=1
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=d4ff93cccb165295ed8357fe42082208b781a3b576bbb7d3fe7a624270b047
b9
```

Java

```
//init request with orderId
GetOrder request = new GetOrder(6642038L);
//fire request and handle result Order
try {
  GetOrderResponse response = request.execute();
  System.out.println(response.getBody());
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Response

Name	Type	Description
OrderId	long	Identifier of this order as assigned by the Seller Center.
CustomerFirstName	string	The customer's first name.
CustomerLastName	string	The customer's last name
OrderNumber	long	The human-readable order number.
PaymentMethod	string	The method of payment.
Remarks	string	A human-readable remark.
DeliveryInfo	string	Order delivery information.
Price	float	Total amount for this order.
GiftOption	boolean	1 if item is a gift, and 0 if it is not.
GiftMessage	string	Gift message as specified by the customer.
CreatedAt	datetime	Date and time when the order was placed.
UpdatedAt	datetime	Date and time of the last change to the order.
AddressBilling	subsection	Node that contains additional nodes, which makes up the billing address: FirstName, LastName, Phone, Phone2, Address1, Address2, City, PostCode, and Country.
AddressShipping	subsection	Node that contains additional nodes, which makes up the shipping address: FirstName, LastName, Phone, Phone2, Address1, Address2, City, PostCode, and Country.
NationalRegistrationNumber	string	Required in some countries.
ExtraAttributes	string	Extra attributes which were passed to the Seller Center on getMarketPlaceOrders call. It is JSON string which client should parse it.
ItemsCount	integer	Number of items in the order.
Statuses	array	Unique status of the items in the order (hint: you can find all of the different status codes in the response example).

TaxCode	string	(For Thailand and Vietnam only) The customer's VAT tax code, provided by the customer when placing the order.
BranchNumber	string	(For Thailand only) The tax branch code for corporate customers, provided by the customer when placing the order.

Result sample

A success result sample is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
     <Head>
          <RequestId/>
          <RequestAction>GetOrder</RequestAction>
          <ResponseType>Order</ResponseType>
          <Timestamp>2015-08-03T11:16:14+0200</Timestamp>
     </Head>
     <Body>
          <Orders>
               <Order>
                    <OrderId>190</OrderId>
                    <CustomerFirstName>First Name
Customer</CustomerFirstName>
                    <CustomerLastName/>
                    <OrderNumber>300125759</OrderNumber>
                    <PaymentMethod>CashOnDelivery</PaymentMethod>
                    <Remarks/>
                    <DeliveryInfo/>
                    <Price>100.00</Price>
                    <GiftOption>0</GiftOption>
                    <GiftMessage/>
                    <VoucherCode/>
                    <CreatedAt>2015-03-24 16:09:22</CreatedAt>
                    <UpdatedAt>2015-03-24 16:09:22</UpdatedAt>
                    <AddressBilling>
                         <FirstName>First Name 3 Test</FirstName>
                         <LastName>First Name 3 Test</LastName>
                         <Phone>0925090880</Phone>
                         <Phone2/>
                         <Address1>Configure this street and
number</Address1>
                         <Address2/>
                         <Address3/>
                         <Address4/>
                         <Address5/>

<CustomerEmail>hello@sellercenter.net</CustomerEmail>
                         <City>Configure city</City>
```

```
                        <Ward/>
                        <Region/>
                        <PostCode>33333</PostCode>
                        <Country>Pakistan</Country>
                    </AddressBilling>
                    <AddressShipping>
                        <FirstName>??????? ???????????</FirstName>
                        <LastName/>
                        <Phone>0925090880</Phone>
                        <Phone2/>
                        <Address1>121 ???? 4 </Address1>
                        <Address2>???????????????????</Address2>
                        <Address3/>
                        <Address4/>
                        <Address5/>

<CustomerEmail>hello@sellercenter.net</CustomerEmail>
                        <City>?????????????-??????-10400</City>
                        <Ward/>
                        <Region/>
                        <PostCode>10400</PostCode>
                        <Country>Pakistan</Country>
                    </AddressShipping>
                    <NationalRegistrationNumber/>
                    <ItemsCount>12</ItemsCount>

<ExtraAttributes>{"TaxInvoiceRequested":"true"}</ExtraAttributes>
                    <Statuses>
                        <Status>shipped</Status>
                        <Status>returned</Status>
                        <Status>return_waiting_for_approval</Status>
                        <Status>return_shipped_by_customer</Status>
                        <Status>return_rejected</Status>
                        <Status>ready_to_ship</Status>
                        <Status>processing</Status>
                        <Status>pending</Status>
                        <Status>failed</Status>
                        <Status>delivered</Status>
                        <Status>canceled</Status>
                    </Statuses>
                    <TaxCode>1234</TaxCode>
                    <BranchNumber>2222</BranchNumber>
                </Order>
```

```
            </Orders>
        </Body>
    </SuccessResponse>
```

Error messages

Error code	Message
16	E016: "%s" Invalid Order ID

GetOrders

Definition

Use this call to get the customer details for a range of orders.

Request URI: https://api.sellercenter.daraz.pk?Action=GetOrders

Parameters

Field	Type	Description
Action	string	*GetOrders* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
CreatedAfter	datetime	Limits the returned orders to those created after or on the specified date, given in ISO 8601 date format. Either CreatedAfter or UpdatedAfter is mandatory.
CreatedBefore	datetime	Limits the returned orders to those created before or on the specified date, given in ISO 8601 date format. Optional
UpdatedAfter	datetime	Limits the returned orders to those updated after or on the specified date, given in ISO 8601 date format. Either UpdatedAfter or CreatedAfter is mandatory.
UpdatedBefore	datetime	Limits the returned orders to those updated before or on the specified date, given in ISO 8601 date format. Optional.
Limit	integer	The maximum number of orders that can be returned. The supported maximum number is 100.
Offset	integer	Number of orders to skip at the beginning of the list (i.e., an offset into the result set; together with the Limit parameter, simple result set paging is possible; if you do page through results, note that the list of orders might change during paging).
Status	string	When set, limits the returned set of orders to loose orders, which return only entries which fit the status provided. Possible values are pending, canceled, ready_to_ship, delivered, returned, shipped and failed.
SortBy	string	Allows to choose the sorting column. Possible values are created_at and updated_at.
SortDirection	string	Specify the sorting type. Possible values are ASC and DESC.

Code sample

Text

```
# GetOrders cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=GetOrders
data-urlencode CreatedAfter=2014-01-01T00%3A00%3A00%2B0200
data-urlencode CreatedBefore=2014-12-31T23%3A59%3A59%2B0200
data-urlencode Offset=0
data-urlencode Limit=100
data-urlencode Status=pending
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=d4ff93cccb165295ed8357fe42082208b781a3b576bbb7d3fe7a624270b047
b9
```

Java

```
//construct filter condition
OrdersFilter filter = new OrdersFilter();
filter.setSortBy(OrdersFilter.SortBy.UpdatedAt)
  .setStatus(OrdersFilter.Status.Canceled)
  .setCreatedAfter(new Date(116,1,1))
  .setSortDirection(OrdersFilter.SortDirection.DESC)
  .setLimit(100);

//init request with filter
GetOrders request = new GetOrders(filter);
//fire request and handle orders
GetOrdersResponse response = request.execute();
for (Order order: response.getBody()) {
  System.out.println(order);
}
```

Response

Name	Type	Description

TotalCount	unsigned	Displayed in the Head section, this number tells the complete number of all orders for the current filter set in the database.
OrderId	long	Identifier of this order as assigned by the Seller Center.
CustomerFirstName	string	The customer's first name.
CustomerLastName	string	The customer's last name.
OrderNumber	long	The human-readable order number.
PaymentMethod	string	The method of payment.
Remarks	string	A human-readable remark.
DeliveryInfo	string	Delivery information for the order.
Price	float	Total amount for this order.
GiftOption	boolean	1 if item is a gift, and 0 if it is not.
GiftMessage	string	Gift message as specified by the customer.
CreatedAt	datetime	Date and time when the order was placed.
UpdatedAt	datetime	Date and time of the last change to the order.
AddressBilling	subsection	Node that contains additional nodes, which makes up the billing address: FirstName, LastName, Phone, Phone2, Address1, Address2, City, PostCode, and Country.
AddressShipping	subsection	Node that contains additional nodes, which makes up the shipping address: FirstName, LastName, Phone, Phone2, Address1, Address2, City, PostCode, and Country.
NationalRegistrationNumber	string	Required in some countries.
ItemsCount	integer	Number of items in order.
Statuses	array	Unique status of the items in the order. (hint: you can find all of the different status codes in the response example)
PromisedShippingTimes	datetime	Target shipping time for the soonest order item if they are available.
ExtraAttributes	string	Extra attributes which were passed to the Seller Center on getMarketPlaceOrders call. It is JSON string which client should parse it.
Voucher	float	Total voucher for this order.
ShippingFee	float	Total shipping fee for this order.
TaxCode	string	The customer's VAT tax code, provided by the customer when placing the order.
BranchNumber	string	The tax branch code for corporate customers, provided by the customer when placing the order.

Result sample

A success result sample is as follows.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
```

```
    <RequestId></RequestId>
    <RequestAction>GetOrders</RequestAction>
    <ResponseType>Orders</ResponseType>
    <Timestamp>2013-08-27T14:44:13+0000</Timestamp>
    <TotalCount>15</TotalCount>
  </Head>
  <Body>
    <Orders>
      <Order>
        <OrderId>1</OrderId>
        <CustomerFirstName>John</CustomerFirstName>
        <CustomerLastName>Doe</CustomerLastName>
        <OrderNumber>3000</OrderNumber>
        <PaymentMethod>CashOnDelivery</PaymentMethod>
        <Remarks></Remarks>
        <DeliveryInfo></DeliveryInfo>
        <Price>100.00</Price>
        <GiftOption>0</GiftOption>
        <GiftMessage></GiftMessage>
        <CreatedAt>2013-09-02 02:28:17</CreatedAt>
        <UpdatedAt>2013-09-02 02:28:17</UpdatedAt>
        <AddressBilling>
          <FirstName>John</FirstName>
          <LastName>Doe</LastName>
          <Phone>0123456789</Phone>
          <Phone2></Phone2>
          <Address1>testtestcarmen</Address1>
          <Address2>testtestcarmen</Address2>
          <CustomerEmail>hello@sellercenter.net</CustomerEmail>
          <City>Kuala Lumpur</City>
          <PostCode>12345</PostCode>
          <Country>Germany</Country>
        </AddressBilling>
        <AddressShipping>
          <FirstName>John</FirstName>
          <LastName>Doe</LastName>
          <Phone>0123456789</Phone>
          <Phone2></Phone2>
          <Address1>testtestcarmen</Address1>
          <Address2>testtestcarmen</Address2>
          <CustomerEmail>hello@sellercenter.net</CustomerEmail>
          <City>Kuala Lumpur</City>
          <PostCode>11111</PostCode>
          <Country>Malaysia</Country>
        </AddressShipping>
        <NationalRegistrationNumber/>
        <ItemsCount>12</ItemsCount>
        <PromisedShippingTime>2015-06-13 17:35:22</PromisedShippingTime>

<ExtraAttributes>{"TaxInvoiceRequested":"true"}</ExtraAttributes>
```

```
  <Statuses>
    <Status>shipped</Status>
    <Status>returned</Status>
    <Status>return_waiting_for_approval</Status>
    <Status>return_shipped_by_customer</Status>
    <Status>return_rejected</Status>
    <Status>ready_to_ship</Status>
    <Status>processing</Status>
    <Status>pending</Status>
    <Status>failed</Status>
    <Status>delivered</Status>
    <Status>canceled</Status>
  </Statuses>
  <Voucher>0</Voucher>
  <ShippingFee>0</ShippingFee>
  <TaxCode>1234</TaxCode>
  <BranchNumber>2222</BranchNumber>
</Order>
```

```
    </Orders>
  </Body>
</SuccessResponse>
```

Error messages

Error code	Message
14	E014: "%s" Invalid Offset
17	E017: "%s" Invalid Date Format
19	E019: "%s" Invalid Limit
36	E036: Invalid status filter
74	E074: Invalid sort direction.
75	E075: Invalid sort filter.

GetOrderItems

Definition

Use this call to get the item information of an order.

Request URI: https://api.sellercenter.daraz.pk?Action=GetOrderItems

Parameters

Field	Type	Description
Action	string	*GetOrderItems* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
OrderId	long	The identifier that was assigned to the order by the Seller Center.

Code sample

Text

```
# GetOrderItems cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=GetOrderItems
data-urlencode OrderId=1
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=d4ff93cccb165295ed8357fe42082208b781a3b576bbb7d3fe7a624270b047
b9
```

Java

```
GetOrderItems request = new GetOrderItems(6642038L);
try {
  GetOrderItemsResponse response = request.execute();
  for (OrderItem orderItem: response.getBody()) {
    System.out.println(orderItem);
  }
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Response

Name	Type	Description
OrderItemId	long	Order item ID
ShopId	long	Seller name
OrderId	long	Order ID
Name	string	Product name
Sku	string	Product SKU
ShopSku	string	Product outer ID in the front-end system
ShippingType	string	1. Drop-shipping 2. Warehouse
ItemPrice	bigdecimal	Product price
PaidPrice	bigdecimal	Paid price = item price - voucher
Currency	string	ISO 4217 compatible currency code

WalletCredits	string	Not used
TaxAmount	bigdecimal	Tax amount
ShippingAmount	bigdecimal	Shipping fee
ShippingServiceCost	bigdecimal	Shipping service cost
VoucherAmount	bigdecimal	Voucher
Status	string	1. Pending 2. ReadyToShip 3. Shipped 4. Delivered 5. Failed_Delivery 6. Return
ShipmentProvider	string	3PL shipment provider, such as DEX
IsDigital	bool	Is digital goods or not
DigitalDeliveryInfo	string	Not used
TrackingCode	string	Tracking code retrieved from 3PL shipment provider
TrackingCodePre	string	Not used
Reason	string	Cancel, Return or other reason, defined in the table sales_order_reason
ReasonDetail	string	Reason detail
PurchaseOrderId	long	Returned by OMS when calling SetPackedByMarketPlace
PurchaseOrderNumber	string	Returned by Lel system when calling SetPackedByMarketPlace
PackageId	string	Package source id, returned by Lel system
PromisedShippingTime	datetime	Not used
ExtraAttributes	string JSON	JSON encoded string with extra attributes
ShippingProviderType	string	One of the following options: Express, Standard, or Economy
CreatedAt	datetime	Time of the feed's creation in ISO 8601 format
UpdatedAt	datetime	Time of the feed's last update in ISO 8601 format

Result sample

A success result sample is as follows.

XML

```
<SuccessResponse>
  <Head>
    <RequestId>0a1430171488289030973215le</RequestId>
    <RequestAction>GetOrderItems</RequestAction>
    <ResponseType>OrderItems</ResponseType>
    <Timestamp>2017-02-28T21:37:11+08:00</Timestamp>
  </Head>
  <Body>
    <OrderItems>
```

```
      <OrderItem>
        <OrderItemId>100967189</OrderItemId>
        <ShopId>migrationtest2</ShopId>
        <OrderId>100663397</OrderId>
        <Name>Test please do not shoot a sku</Name>
        <Sku>545252329001-3452605530003</Sku>
        <ShopSku>OE702FAAA7HM6ASGAMZ-14920937</ShopSku>
        <ShippingType>Dropshipping</ShippingType>
        <ItemPrice>4.40</ItemPrice>
        <PaidPrice>4.40</PaidPrice>
        <Currency>SGD</Currency>
        <WalletCredits>0.00</WalletCredits>
        <TaxAmount>0.00</TaxAmount>
        <ShippingAmount>1.00</ShippingAmount>
        <ShippingServiceCost>0</ShippingServiceCost>
        <VoucherAmount>0.00</VoucherAmount>
        <VoucherCode>
        </VoucherCode>
        <Status>pending</Status>
        <ShipmentProvider>LGS-FM50</ShipmentProvider>
        <IsDigital>0</IsDigital>
        <DigitalDeliveryInfo>
        </DigitalDeliveryInfo>
        <TrackingCode>LZD5000000000311SG</TrackingCode>
        <TrackingCodePre>
        </TrackingCodePre>
        <Reason>
        </Reason>
        <ReasonDetail>
        </ReasonDetail>
        <PurchaseOrderId>1981744</PurchaseOrderId>

<PurchaseOrderNumber>MPDS-M170861324375076219444</PurchaseOrderNumber>
        <PackageId>MPDS-303172379-1927</PackageId>
        <PromisedShippingTime>
        </PromisedShippingTime>
        <ExtraAttributes>null</ExtraAttributes>
        <ShippingProviderType>standard</ShippingProviderType>
        <CreatedAt>2017-02-26 15:52:00</CreatedAt>
        <UpdatedAt>2017-02-28 21:35:04</UpdatedAt>
        <ReturnStatus>
        </ReturnStatus>

<productMainImage>http://sg-live-02.slatic.net/p/7/test-please-do-not-sh
oot-a-sku-3703-20297521-a6ee04ac9d93d572c22b92d3a3a00afa-catalog.jpg</pr
oductMainImage>
        <Variation>Not Specified</Variation>

<ProductDetailUrl>http://www.daraz.pk/12579202.html</ProductDetailUrl>
      </OrderItem>
```

```
      <OrderItem>
        <OrderItemId>100967191</OrderItemId>
        <ShopId>migrationtest2</ShopId>
        <OrderId>100663397</OrderId>
        <Name>Test please do not shoot a sku</Name>
        <Sku>545252329001-3452605530003</Sku>
        <ShopSku>OE702FAAA7HM6ASGAMZ-14920937</ShopSku>
        <ShippingType>Dropshipping</ShippingType>
        <ItemPrice>4.40</ItemPrice>
        <PaidPrice>4.40</PaidPrice>
        <Currency>SGD</Currency>
        <WalletCredits>0.00</WalletCredits>
        <TaxAmount>0.00</TaxAmount>
        <ShippingAmount>1.00</ShippingAmount>
        <ShippingServiceCost>0</ShippingServiceCost>
        <VoucherAmount>0.00</VoucherAmount>
        <VoucherCode>
        </VoucherCode>
        <Status>pending</Status>
        <ShipmentProvider>LGS-FM50</ShipmentProvider>
        <IsDigital>0</IsDigital>
        <DigitalDeliveryInfo>
        </DigitalDeliveryInfo>
        <TrackingCode>LZD5000000000311SG</TrackingCode>
        <TrackingCodePre>
        </TrackingCodePre>
        <Reason>
        </Reason>
        <ReasonDetail>
        </ReasonDetail>
        <PurchaseOrderId>1981744</PurchaseOrderId>

<PurchaseOrderNumber>MPDS-M170861324375076219444</PurchaseOrderNumber>
        <PackageId>MPDS-303172379-1927</PackageId>
        <PromisedShippingTime>
        </PromisedShippingTime>
        <ExtraAttributes>null</ExtraAttributes>
        <ShippingProviderType>standard</ShippingProviderType>
        <CreatedAt>2017-02-26 15:52:00</CreatedAt>
        <UpdatedAt>2017-02-28 21:35:04</UpdatedAt>
        <ReturnStatus>
        </ReturnStatus>

<productMainImage>http://sg-live-02.slatic.net/p/7/test-please-do-not-sh
oot-a-sku-3703-20297521-a6ee04ac9d93d572c22b92d3a3a00afa-catalog.jpg</pr
oductMainImage>
        <Variation>Not Specified</Variation>

<ProductDetailUrl>http://www.daraz.pk/12579202.html</ProductDetailUrl>
      </OrderItem>
```

```
      <OrderItem>
        <OrderItemId>100967192</OrderItemId>
        <ShopId>migrationtest2</ShopId>
        <OrderId>100663397</OrderId>
        <Name>Test please do not shoot a sku</Name>
        <Sku>545252329001-3452605530003</Sku>
        <ShopSku>OE702FAAA7HM6ASGAMZ-14920937</ShopSku>
        <ShippingType>Dropshipping</ShippingType>
        <ItemPrice>4.40</ItemPrice>
        <PaidPrice>4.40</PaidPrice>
        <Currency>SGD</Currency>
        <WalletCredits>0.00</WalletCredits>
        <TaxAmount>0.00</TaxAmount>
        <ShippingAmount>1.00</ShippingAmount>
        <ShippingServiceCost>0</ShippingServiceCost>
        <VoucherAmount>0.00</VoucherAmount>
        <VoucherCode>
        </VoucherCode>
        <Status>pending</Status>
        <ShipmentProvider>LGS-FM50</ShipmentProvider>
        <IsDigital>0</IsDigital>
        <DigitalDeliveryInfo>
        </DigitalDeliveryInfo>
        <TrackingCode>LZD5000000000311SG</TrackingCode>
        <TrackingCodePre>
        </TrackingCodePre>
        <Reason>
        </Reason>
        <ReasonDetail>
        </ReasonDetail>
        <PurchaseOrderId>1981744</PurchaseOrderId>

<PurchaseOrderNumber>MPDS-M170861324375076219444</PurchaseOrderNumber>
        <PackageId>MPDS-303172379-1927</PackageId>
        <PromisedShippingTime>
        </PromisedShippingTime>
        <ExtraAttributes>null</ExtraAttributes>
        <ShippingProviderType>standard</ShippingProviderType>
        <CreatedAt>2017-02-26 15:52:00</CreatedAt>
        <UpdatedAt>2017-02-28 21:35:04</UpdatedAt>
        <ReturnStatus>
        </ReturnStatus>

<productMainImage>http://sg-live-02.slatic.net/p/7/test-please-do-not-sh
oot-a-sku-3703-20297521-a6ee04ac9d93d572c22b92d3a3a00afa-catalog.jpg</pr
oductMainImage>
        <Variation>Not Specified</Variation>

<ProductDetailUrl>http://www.daraz.pk/12579202.html</ProductDetailUrl>
      </OrderItem>
```

```
    </OrderItems>
  </Body>
</SuccessResponse>
```

Error messages

Error code	Message
16	E016: "%s" Invalid Order ID

GetMultipleOrderItems

Definition

Use this call to get the item information of one or more orders.

Request URI: https://api.sellercenter.daraz.pk?Action=GetMultipleOrderItems

Parameters

Field	Type	Description
Action	string	*GetMultipleOrderItems* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	DateTime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
OrderId	long	Comma-separated list of order identifiers in square brackets.

Code sample, response, and result sample

The code sample, response, and result sample are similar with those of the *GetOrderItems* call.

Error messages

Error code	Message
37	E037: One or more order id in the list are incorrect
38	E038: Too many orders were requested
39	E039: No orders were found
56	E056: Invalid OrdersIdList format. Must use array format [1,2]

SetInvoiceNumber

Definition [POST]

Use this call to set the invoice access key.

Request URI: https://api.sellercenter.daraz.pk?Action=SetInvoiceNumber

Parameters

Field	Type	Description
Action	string	*SetInvoiceNumber* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	DateTime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
OrderId	long	Identifier of the order item.
InvoiceNumber	string	The invoice number.

Code sample

Text

```
# OrderItemUpdate cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=SetInvoiceNumber
data-urlencode OrderItemId=1
data-urlencode
InvoiceNumber=12345678912345678912345678912345678912345678
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=d4ff93cccb165295ed8357fe42082208b781a3b576bbb7d3fe7a624270b047
b9
```

Java

```
//init request with invoice number and order item id
SetInvoiceNumber request = new SetInvoiceNumber("12349989", 102612420L);
//fire request and check result
try {
  UpdateResponse response = request.execute();
  System.out.println("SetInvoiceNumber succeed?" +
response.isSucceed());
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Result sample

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId/>
    <RequestAction>SetInvoiceNumber</RequestAction>
    <ResponseType/>
    <Timestamp>2015-06-04T09:35:37+0200</Timestamp>
  </Head>
  <Body>
      <OrderItemId>1</OrderItemId>
      <InvoiceNumber>INV-20</InvoiceNumber>
  </Body>
</SuccessResponse>
```

Documentation

> **Note**
> Invoice will be changed for all order items in the package.

SetStatusToPackedByMarketplace

Definition [POST]

Use this call to mark order items as being packed.

Request URI: https://api.sellercenter.daraz.pk?Action=SetStatusToPackedByMarketplace

Parameters

Field	Type	Description
Action	string	*SetStatusToPackedByMarketplace* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	DateTime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
OrderItemIds	array of long	List of order items to be marked ready to ship. Comma separated list in square brackets. Mandatory.
DeliveryType	string	One of the following delivery types: 'dropship' - the seller will send out the package on his own. 'pickup' - shop should pick up the item from the seller (cross-docking). 'send_to_warehouse' - the seller will send the item to the warehouse (cross-docking).
ShippingProvider	string	Valid shipment provider as looked up via *GetShipmentProviders*. Leave this null (blank) if shipment provider is assigned to you during order processing.

Code sample

Text

```
# SetStatusToPackedByMarketplace cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=SetStatusToPackedByMarketplace
data-urlencode OrderItemIds=[1]
data-urlencode DeliveryType=dropship
data-urlencode ShippingProvider=Aramax
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=d4ff93cccb165295ed8357fe42082208b781a3b576bbb7d3fe7a624270b047
b9
```

Java

```
//init request with OrderItemIds, DeliverType, ShipmentProvider
//for ShipmentProvider, you can get options by calling
GetShipmentProvders
List<Long> ids = Arrays.asList(102433828L, 102367937L);
SetStatusToPackedByMarketplace request = new
SetStatusToPackedByMarketplace(ids,

DeliveryType.Dropship, "LEX");
//fire request and handle result
try {
  PackedByMarketPlaceResponse response = request.execute();
  for(PackageInfo packageInfo: response.getBody()) {
    System.out.println(packageInfo);
  }
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Multiple items
For packing multiple items in the same package, input order items in OrderItemIds parameter as comma separated values enclosed in brackets. For e.g: OrderItemIds=[102352123451,102352125324]. Order number should be same.

Response

The API responds by sending purchase order details in the response.

Name	Type	Description
OrderItemId	long	Seller Center order item ID
PurchaseOrderId	long	OMS order ID
PurchaseOrderNumber	string	OMS order number
PackageId	string	Shipping package ID
TrackingNumber	string	Package tracking number

Result sample

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId></RequestId>
    <RequestAction>SetStatusToPackedByMarketplace</RequestAction>
    <ResponseType>OrderItems</ResponseType>
    <Timestamp>2013-08-27T14:44:13+0000</Timestamp>
  </Head>
  <Body>
    <OrderItems>
      <OrderItem>
        <OrderItemId>1</OrderItemId>
        <PurchaseOrderId>123456</PurchaseOrderId>
        <PurchaseOrderNumber>ABC-123456</PurchaseOrderNumber>
        <PackageId>MPDS-200131783-9800</PackageId>
        <ShipmentProvider>5</ShipmentProvider>
        <TrackingNumber>TRACK-1126935-4306</TrackingNumber>
      </OrderItem>
    </OrderItems>
  </Body>
</SuccessResponse>
```

Error messages

Error code	Message
20	E020: "%s" Invalid Order Item ID
21	E021: OMS Api Error Occurred
23	E023: "%s" Invalid Order Item IDs
24	E024: "%s" Invalid Delivery Type
25	E025: "%s" Invalid Shipping Provider
29	E029: Order items must be from the same order
73	E073: All order items must have status Pending or Ready To Ship. (%s)
91	E091: You are not allowed to set the shipment provider and tracking number and the delivery type is wrong. Please use sent_to_warehouse

---

SetStatusToReadyToShip

Definition

Use this call to mark an order item as being ready to ship.

Request URI: https://api.sellercenter.daraz.pk?Action=SetStatusToReadyToShip

Prerequisite
Call the SetStatusToPackedByMarketPlace API before calling SetStatusToReadyToShip.

Parameters

Field	Type	Description
Action	string	*SetStatusToReadyToShip* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	DateTime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
OrderItemIds	array of long	List of order items to be marked ready to ship. Comma separated list in square brackets. Mandatory
DeliveryType	string	One of the following delivery types: 'dropship' - the seller will send out the package on his own. 'pickup' - shop should pick up the item from the seller (cross-docking). 'send_to_warehouse' - the seller will send the item to the warehouse (cross-docking). Mandatory. Now only "dropship" is supported.
ShippingProvider	string	Valid shipment provider as looked up via *GetShipmentProviders*. Leave this null (blank) if shipment provider is assigned to you during order processing.
TrackingNumber	string	Package tracking number. Mandatory in the case of drop-shipping.
SerialNumber	mixed	Only applicable if the serial number feature has been enabled for platform and seller. Unique serial number for tracking order items, which can be specified as comma-separated list of serials or JSON array, in which case there must be exactly as many serials as there are OrderItemIds. Can also be specified as a JSON object where keys are order item ids and values are corresponding serial numbers. In this case the order item ID from OMS will be used as serial for items where it has not been explicitly specified. Optional.

Code sample

Text

```
# SetStatusToReadyToShip cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=SetStatusToReadyToShip
data-urlencode OrderItemIds=[1]
data-urlencode DeliveryType=dropship
data-urlencode ShippingProvider=Aramax
data-urlencode TrackingNumber=123456789
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=d4ff93cccb165295ed8357fe42082208b781a3b576bbb7d3fe7a624270b047
b9
```

Java

```
//init request with OrderItemId list, DeliveryType, ShippingProvider,
TrackingNumber
List<Long> orderItemIds = Arrays.asList(102612420L, 102612419L);
SetStatusToReadyToShip request = new
SetStatusToReadyToShip(orderItemIds, DeliveryType.Dropship, "SkyNet -
DS", "23873905618");
try {
  UpdateResponse response = request.execute();
  System.out.println("Update status succeed?" + response.getBody());
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Multiple items
To RTS multiple items in the same package, input order items in OrderItemIds parameter as comma separated values enclosed in brackets. For e.g: OrderItemIds=[102352123451,102352125324]. Order number should be same.

Response

The API responds by sending purchase order details in the response:

Name	Type	Description
PurchaseOrderId	long	Seller Center identification. Optional, please ignore it if your business scenario does not cover it.

PurchaseOrderNumber	string	Order number in the Seller Center. Optional, please ignore it if your business scenario does not cover it.

Result sample

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId></RequestId>
    <RequestAction>SetStatusToReadyToShip</RequestAction>
    <ResponseType>OrderItems</ResponseType>
    <Timestamp>2013-08-27T14:44:13+0000</Timestamp>
  </Head>
  <Body>
    <OrderItems>
      <OrderItem>
        <PurchaseOrderId>123456</PurchaseOrderId>
        <PurchaseOrderNumber>ABC-123456</PurchaseOrderNumber>
      </OrderItem>
    </OrderItems>
  </Body>
</SuccessResponse>
```

Error messages

Error code	Message	Explanation
20	E020: "%s" Invalid Order Item ID	
21	E021: OMS Api Error Occurred	
23	E023: "%s" Invalid Order Item IDs	
24	E024: "%s" Invalid Delivery Type	
25	E025: "%s" Invalid Shipping Provider	
26	E026: "%s" Invalid Tracking Number	
29	E029: Order items must be from the same order	
31	E031: Tracking ID incorrect. Example tracking ID: "%s"	
63	E063: The tracking code %s has already been used	
73	E073: All order items must have status Pending or Ready To Ship. (%s)	
91	E091: You are not allowed to set the shipment provider and tracking number and the delivery type is wrong. Please use sent_to_warehous	
94	E094: Serial numbers specified incorrectly	Serial numbers were not specified according to one of the accepted formats for the `SerialNumber` parameter.

95	E095: Invalid serial number format (%s)	Serial numbers must be 1 to 26 characters; only latin letters and digits are allowed.
96	E096: Duplicate serial number among order items (%s)	Two or more items in the order would share a serial number.

GetDocument

Definition

Use this call to retrieve order-related documents, including invoices, shipping labels, and shipping parcels.

Request URI: https://api.sellercenter.daraz.pk?Action=GetDocument

Parameters

Field	Type	Description
Action	string	*GetDocument* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	DateTime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
DocumentType	string	Document types, including 'invoice', 'shippingLabel', or 'carrierManifest'. Mandatory.
OrderItemIds	array of long	Identifier of the order item for which the caller wants to get a document. Mandatory.

Code sample

Text

```
# GetDocument cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=GetDocument
data-urlencode DocumentType=shippingLabel
data-urlencode OrderItemIds=[1,2]
data Timestamp=2015-07-14T04%3A27%2B0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=4bef3fda564dd24173260066b97c4e4af08dc9984a0ce147de2f476c167e64
75
```

Java

```
List<Long> ids = Arrays.asList(102612420L,102612419L);
GetDocument GetDocument = new GetDocument(DocumentType.ShippingLabel,
ids);
try {
  GetDocumentResponse response = GetDocument.execute();
  Document document = response.getBody();
  System.out.println(document);
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Result sample

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId></RequestId>
    <RequestAction>GetDocument</RequestAction>
    <ResponseType></ResponseType>
    <Timestamp>2013-08-27T14:44:13+0000</Timestamp>
  </Head>
  <Body>
    <Document>
      <DocumentType>shippingLabel</DocumentType>
      <MimeType>text/html</MimeType>
      <File>YTM0NZomIzI2OTsmIzM0NTueYQ==</File>
    </Document>
  </Body>
</SuccessResponse>
```

Documentation

To reconstruct the file, the data from the <File> node needs to be base64 decoded, and interpreted according to the <MimeType>.

Error messages

Error code	Message
20	E020: "%s" Invalid Order Item IDs
21	E021: OMS Api Error Occurred
32	E032: Document type "%s" is not valid
34	E034: Order Item must be packed. Please call setStatusToReadyToShip before
35	E035: "%s" was not found

GetFailureReasons

Definition

Use this call to get additional error context for *SetStatusToCanceled*.

Request URI: https://api.sellercenter.daraz.pk?Action=GetFailureReasons

Parameters

Field	Type	Description
Action	string	*GetFailureReasons* Name of the API that is to be called. Mandatory.

Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	DateTime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Code sample

Text

```
# GetFailureReasons cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=GetFailureReasons
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=d4ff93cccb165295ed8357fe42082208b781a3b576bbb7d3fe7a624270b047
b9
```

Java

```
GetFailureReasons request = new GetFailureReasons();
try {
  GetFailureReasonsResponse response = request.execute();
  for (FailureReason reason: response.getBody()) {
    System.out.println(reason);
  }
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Response

The reason for each failure is given inside the following elements of the XML response.

Name	Type	Description
Type	string	Type of failure. Possible values are 'canceled', 'failed' and 'returned'.

ReasonId	long	ID of the failure reason, which is used by *SetStatusToCanceled*.
Name	string	Short description of the failure reason.

Result sample

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId></RequestId>
    <RequestAction>GetFailureReasons</RequestAction>
    <ResponseType>Reasons</ResponseType>
    <Timestamp>2013-08-27T14:44:13+0000</Timestamp>
  </Head>
  <Body>
    <Reasons>
      <Reason>
        <Type>canceled</Type>
        <Name>Out of stock</Name>
        <ReasonId>15</ReasonId>
      </Reason>
      <Reason>
        <Type>canceled</Type>
        <Name>Wrong Price or Pricing Error</Name>
        <ReasonId>21</ReasonId>
      </Reason>

    </Reasons>
  </Body>
</SuccessResponse>
```

Error messages

No specific errors.

SetStatusToCanceled

Definition [POST]

Use this call to cancel a single item.

Request URI: https://api.sellercenter.daraz.pk?Action=SetStatusToCanceled

Parameters

Field	Type	Description

Action	string	*SetStatusToCanceled* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	DateTime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
OrderItemId	long	Order item ID. Mandatory.
ReasonId	long	ID of the cancel reason, which is returned by the *GetFailureReasons* API. Mandatory.
ReasonDetail	string	Reason detail. Optional.

Code sample

Text

```
# SetStatusToCanceled cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=SetStatusToCanceled
data-urlencode OrderItemId=1
data-urlencode ReasonID=15
data-urlencode ReasonDetail=My hobbies are ordering and returning.
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=d4ff93cccb165295ed8357fe42082208b781a3b576bbb7d3fe7a624270b047
b9
```

Java

```
//init request with OrderItemId, reason, detailed reason
SetStatusToCanceled request = new SetStatusToCanceled(10937342L,
"Invalid - Customer unreachable", "detailreason");
//fire request and check result
try {
  UpdateResponse response = request.execute();
  System.out.println("Set status succeed?" + response.getBody());
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Result sample

Success and failure result samples are as follows.

Success

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId></RequestId>
    <RequestAction>SetStatusToCanceled</RequestAction>
    <ResponseType></ResponseType>
    <Timestamp>2013-08-27T14:44:13+0000</Timestamp>
  </Head>
  <Body />
</SuccessResponse>
```

Failure

```
<?xml version="1.0" encoding="UTF-8"?>
<ErrorResponse>
  <Head>
    <RequestAction>SetStatusToCanceled</RequestAction>
    <ErrorType>Sender</ErrorType>
    <ErrorCode>1000</ErrorCode>
    <ErrorMessage>Invalid cancelation reason</ErrorMessage>
  </Head>
  <Body/>
</ErrorResponse>
```

Error messages

Error code	Message
20	E020: "%s" Invalid Order Item ID
21	E021: OMS Api Error Occurred
22	E022: "%s" Invalid Reason
28	E028: It is not possible to set the order to the status "%s"

QUALITY CONTROL ENDPOINTS

GetQcStatus

Definition [GET]

Use this call to get the quality control status of items being listed.

Request URI: https://api.sellercenter.daraz.pk?Action=GetQcStatus

Parameters

Field	Type	Description
Action	string	*GetQcStatus* Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
Limit	integer	The maximum number of QC status entries that can be returned. If you omit this parameter, the default value of 100 is used.
Offset	integer	Number of QC status entries to skip (i.e., an offset into the result set; together with the Limit parameter, simple result set paging is possible; if you do page through results, note that the list of QC status entries might change during paging).
SkuSellerList	array of strings	QC status will be returned for SKUs in this list only. Can either be a JSON array (e.g., [“E213”,”KI21”,”HT0”]) or a serialized PHP array (e.g., a:3:{i:0;s:4:”E213”;i:1;s:4:”KI21”;i:2;s:3:”HT0”;}).

Code sample

Text

```
# GetQcStatus cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=GetQcStatus
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=9ade00fb4b9ab9ed1b8a4d189f1a13e1029edc25dd963235480ec69226fd9f
39
data-urlencode SellerSkuList=["ABC"]
```

Java

```
//init request with list of seller sku
List<String> skus = new ArrayList<>();
skus.add("ASC Test 12 Multisourced - ME");
skus.add("ASC Test 12 Normal - EL");
skus.add("ASC Test 12 Non-COD - EL");
GetQcStatus request = new GetQcStatus(skus);
//fire request and check resualt
try {
  GetQcStatusResponse response = request.execute();
  for(QcStatus qcStatus: response.getBody()) {
    System.out.println(qcStatus);
  }
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Result sample

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
     <Head>
          <RequestId/>
          <RequestAction>GetQcStatus</RequestAction>
          <ResponseType>State</ResponseType>
          <Timestamp>2016-04-28T15:09:01+0200</Timestamp>
     </Head>
     <Body>
          <Status>
               <State>
                    <SellerSKU>TestProduct2030</SellerSKU>
                    <Status>approved</Status>
                    <DataChanged>1</DataChanged>
               </State>
               <State>
                    <SellerSKU>TestProduct2031</SellerSKU>
                    <Status>approved</Status>
                    <DataChanged>1</DataChanged>
               </State>
               <State>
                    <SellerSKU>TestProduct2032</SellerSKU>
                    <Status>pending</Status>
               </State>
               <State>
                    <SellerSKU>TestProduct2033</SellerSKU>
                    <Status>pending</Status>
               </State>
               <State>
                    <SellerSKU>TestProduct2034</SellerSKU>
                    <Status>rejected</Status>
                    <Reason>Wrong Description; Wrong
Translation</Reason>
               </State>
               <State>
                    <SellerSKU>TestProduct2035</SellerSKU>
                    <Status>rejected</Status>
                    <Reason>Price Not Reasonable; Image Corrupt</Reason>
               </State>
          </Status>
     </Body>
</SuccessResponse>
```

Request parameter restrictions
SkuSellerList- Be aware that the parameter can ONLY be sent as an array so in the case you want to use it with only one element it should be still an array with one element.

For example, SkuSellerList = ["E213"].

Sending a string instead of an array will trigger an error.

Expected result
The method will not return old QC results, but only recent ones. Provided you have a live product with several variations approved previously, then you add a new variation and query by all product variation SKUs. Result of such call will consist of 1 entry - a new variation status.

SELLER ENDPOINTS

GetPayoutStatus

Definition

Use this call to get the payout status for a specified period.

Request URI:https://api.sellercenter.daraz.pk?Action=GetPayoutStatus

Parameters

Field	Type	Description
Action	string	GetPayoutStatus Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
CreatedAfter	integer	Filter statements created after the provided date. Mandatory.

Code sample

Text

```
# GetPayoutStatus cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=GetPayoutStatus
data-urlencode CreatedAfter=2015-01-01T11:11+0200
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=d4ff93cccb165295ed8357fe42082208b781a3b576bbb7d3fe7a624270b047
b9
```

Java

```
GetPayoutStatus request = new GetPayoutStatus();
try {
  GetPayoutStatusResponse response = request.execute();
  for(PayoutStatus payoutStatus: response.getBody()) {
    System.out.println(payoutStatus);
  }
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Response

The response contains the following fields

Name	Type	Description
StatementNumber	string	Statement identifier
CreatedAt	datetime	When the statement was created
UpdatedAt	datetime	When the statement was last updated
OpeningBalance	decimal	The opening balance
ItemRevenue	decimal	The revenue generated by the item
ShipmentFee	decimal	Costs of shipping
ShipmentFeeCredit	decimal	Shipping fee credit, if any
OtherRevenueTotal	decimal	Other total revenue
FeesTotal	decimal	Sum of payment fee and return to seller fee

Subtotal1	decimal	Sum of item revenue and other revenue
Refunds	decimal	Sum of all refunds, if any
FeesOnRefundsTotal	decimal	Accumulated fees on refunds issued
Subtotal2	decimal	(Sum of Subtotal1) - Refunds
ClosingBalance	decimal	Closing balance
GuaranteeDeposit	decimal	Guarantee deposit
Payout	decimal	Amount to be paid out to seller for statement
Paid	boolean	Payout status of statement. 1 is paid, and 0 is not paid

Result sample

XML


```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestId/>
    <RequestAction>GetPayoutStatus</RequestAction>
    <ResponseType>Statement</ResponseType>
    <Timestamp>2015-06-16T13:57:59+0000</Timestamp>
  </Head>
  <Body>
    <PayoutStatus>
      <Statement>
        <StatementNumber>EG100RT-20141228</StatementNumber>
        <CreatedAt>2014-01-04 00:23:04</CreatedAt>
        <UpdatedAt>2014-01-04 00:23:04</UpdatedAt>
        <OpeningBalance>0.00</OpeningBalance>
        <ItemRevenue>0</ItemRevenue>
        <ShipmentFee>51.20</ShipmentFee>
        <ShipmentFeeCredit>51.20</ShipmentFeeCredit>
        <OtherRevenueTotal>0</OtherRevenueTotal>
        <FeesTotal>51.20</FeesTotal>
        <Subtotal1>-51.20</Subtotal1>
        <Refunds>0</Refunds>
        <FeesOnRefundsTotal>0</FeesOnRefundsTotal>
        <Subtotal2>-51.20</Subtotal2>
        <ClosingBalance>3962.41</ClosingBalance>
        <GuaranteeDeposit>0</GuaranteeDeposit>
        <Payout>3962.41 EUR</Payout>
        <Paid>0</Paid>
      </Statement>
      <Statement>
        <StatementNumber>EG100RT-20141229</StatementNumber>
        <CreatedAt>2015-01-04 00:23:04</CreatedAt>
        <UpdatedAt>2015-01-04 00:23:04</UpdatedAt>
        <OpeningBalance>10.00</OpeningBalance>
```

```
    <ItemRevenue>0</ItemRevenue>
    <ShipmentFee>1.50</ShipmentFee>
    <ShipmentFeeCredit>1.50</ShipmentFeeCredit>
    <OtherRevenueTotal>0</OtherRevenueTotal>
    <FeesTotal>1.50</FeesTotal>
    <Subtotal1>-1.50</Subtotal1>
    <Refunds>0</Refunds>
    <FeesOnRefundsTotal>0</FeesOnRefundsTotal>
    <Subtotal2>-1.50</Subtotal2>
    <ClosingBalance>777.77</ClosingBalance>
    <GuaranteeDeposit>0</GuaranteeDeposit>
    <Payout>666.66 EUR</Payout>
    <Paid>0</Paid>
  </Statement>
```

```
    </PayoutStatus>
  </Body>
</SuccessResponse>
```

Error messages

Error code	Message
4	E004: Invalid Timestamp format
62	E062 No seller was found by e-mail %s

GetTransactionDetails

Definition

Use this call to get transaction or fee details for a specified period.

Request URI: https://api.sellercenter.daraz.pk?Action=GetTransactionDetails

Parameters

Field	Type	Description
Action	string	GetTransactionDetails Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.
startTime & endTime	date (YYYY-MM-DD)	Starting date and ending date where transactions need to be extracted.
maxItem	integer	Number of maximum lines of transactions to be extracted.
transType	integer	Transaction type ID.
Limit	integer	The maximum number of QC status entries that can be returned. If you omit this parameter, the default value of 100 is used.
Offset	integer	Number of QC status entries to skip (i.e., an offset into the result set; together with the Limit parameter, simple result set paging is possible; if you do page through results, note that the list of QC status entries might change during paging).
SkuSellerList	array of strings	QC status will be returned for SKUs in this list only. Can either be a JSON array (e.g., [“E213”,”KI21”,”HT0”]) or a serialized PHP array (e.g., a:3:{i:0;s:4:”E213”;i:1;s:4:”KI21”;i:2;s:3:”HT0”;}).

Code sample

Text

```
# GetTransactionDetails cURL example
# to run, update Timestamp and recompute Signature
#
url = "https://api.sellercenter.daraz.pk/"
get
data-urlencode Action=GetTransactionDetails
data-urlencode CreatedAfter=2015-01-01T11:11+0200
data-urlencode Timestamp=2015-07-01T11:11+0000
data-urlencode UserID=maintenance@sellercenter.net
data-urlencode Version=1.0
data-urlencode
Signature=d4ff93cccb165295ed8357fe42082208b781a3b576bbb7d3fe7a624270b047
b9
```

Java

```
GetTransactionDetails request = new GetTransactionDetails();
try {
  GetTransactionDetailsResponse response = request.execute();
  for(TransactionDetails transactionDetails: response.getBody()) {
    System.out.println(transactionDetails);
  }
} catch (DarazException e) {
  System.out.println(e.getResponseStr());
}
```

Response

Name	Type	Description
Transaction Date	date (DD Month YYYY)	Date of the transaction
Transaction Type	Transaction type or fee name (integer)	Transaction type or fee name
Transaction Number	Unique id of the transaction in the format Seller code-YYYYYY (string)	ex - SG103EF-1FF0JVW
Amount	integer (decimal)	Total transaction value
WHT included in Amount	boolean	Yes / No
Statement	string	Statement ID
Paid Status	boolean	Yes / No
Order no	string	Order ID
Order Item no	string	Order item number
Shipment Type	string	Dropshipping

Reference	string	Payment reference ID
Comments	string	Comments by regional finance team

Transaction types

Some common transaction types are as follows.

Transaction type	Transaction ID	Purpose
All transactions	-1	Fetching all transactions
Orders - Item charges - Item Price Credit	13	Item price credit is credit provided to successfully delivered items. It's triggered automatically when Order Item status moved to 'delivered'.
Orders - Item charges - Item Price Subsidy	59	Item price subsidy is a credit provided to sellers for an order item.
Orders - Claims - Penalty & Payment Claims	60	Claims (credit) provided to sellers for any penalties or payment related.
Orders - Claims - Other Claims	139	Claims (credit) provided to sellers for other reasons.
Orders - Other Credit - Shipping Fee (Paid By Customer)	8	Shipping fee paid by customers are provided as credit to sellers.
Orders - Other Credit - Other Credit - Taxable	63	Other credit (taxable) for delivered orders.
Orders - Other Credit - Other Credit - Non Taxable	64	Other credit (non-taxable) for delivered orders.
Orders - Other Debit - Non Taxable	144	Other debits (non-taxable) for delivered orders.
Orders - Other Debit - Taxable	82	Other debits (taxable) for delivered orders.
Orders - Daraz Fees - Commissions	16	Commissions for delivered orders.
Orders - Daraz Fees - Payment Fee	3	Payment fee for delivered orders.
Orders - Daraz fees - Shipping Fee (Charged by Daraz)	7	Shipping fee (charged by Daraz) for delivered orders.
Orders - Daraz Fees - Adjustments Shipping Fee	67	Adjustments shipping fee is a credit for delivered orders.
Orders - Daraz Fees - Automatic Shipping Fee	28	Automatic shipping fee is a debit to claw back shipping fee credits.
Orders - Penalties - Fake RTS	71	Penalties for fake RTS.
Orders - Penalties - Infringed Content	76	Penalties for infringed content.
Orders - Penalties - Adjustment penalty	81	Adjustment penalty.
Refunds - Item Charges - Reversal Item Price	14	Reversal item price is an automatic debit for returned items.
Refunds - claims - Lost And Damaged Claim	85	Lost and damaged claim is a credit for sellers for loss/damaged due to 3PL faults.
Refunds - Claims - Other Claims - Returns	125	Other claims (returns) is a credit to sellers for other reasons.
Refunds - Shipping Fee - Reversal shipping Fee (Paid by Customer)	142	Reversal shipping fee (paid by customer) is auto clawback of shipping fee for returned items.
Refunds - Daraz Fees -Reversal Commission	15	Reversal commission is auto reversal of commission for returned items.
Refunds - Daraz Fees - Reversal Shipping Fee (Charged By Daraz)	143	Reversal shipping fee (charged By Daraz) is manual reversal (credit) of shipping fee.
Refunds - penalties - Cancellation Due To OOS	87	Penalties for cancellation of orders due to out of stock items.

Refunds - penalties - Cancellation Due To late shipment	93	Penalties for cancellation due to late shipment.
Refunds - penalties -Forbidden/Infringed Goods	95	Penalties for forbidden/infringed goods.
Refunds - Other Debits (Returns)	145	Other debits for sellers for returned / cancelled items.
Refunds - Adjustments Others (Returns)	104	Other credits for sellers for returned / cancelled items.
3P services - shipping - Pickup fee	127	Pickup fee.
3P services - shipping - Shipping fee (charged by 3P)	21	Shipping fee (charged by 3PL such as LEL) or direct shipping fee post by 3PL to sellers. It's a debit.
3P services - shipping - Return Delivery Fee	24	Return delivery fee charged by 3PL. Its a debit.
3P services - shipping -Adjustments Shipping	129	Any adjustments shipping (credit).
3P services - FBL - FBL Handling Fee	130	FBL handling fee (debits).
3P services - FBL -FBL - Other Fulfillment service Fee	133	Other fulfillment service fee posted by FBL (debit).
3P services - FBL -Adjustments FBL	134	Adjustments FBL is a credit posted to offset any incorrect posting by FBL.
Other services - subsidy - Seller Incentive	107	Seller incentive is a credit to sellers (seller level).
Other services - services Sponsored Product Fee	112	Fees for sponsored product at seller level.
Other services - services Adjustments Others	115	Adjustments others is a credit at seller level to offset any debit posting.
Other services - Other Debits	116	Other debits is a debit at seller level.

Result sample

JSON


```
{
    "SuccessResponse": {
        "Head": {
            "RequestId": "",
            "RequestAction": "GetTransactionDetails",
            "ResponseType": "List",
            "Timestamp": "2017-05-04T19:18:06+0800"
        },
        "Body": {
            "TransactionDOs": {
                "transactionDOs": [
                    {
                        "Transaction Date": "17 May 2016",
                        "Transaction Type": "Shipping Fee (Item Level)
Credit",
                        "Transaction Number": "SG103EF-1KV89DI",
                        "Amount": "10.00",
                        "WHT included in Amount": "Yes",
                        "Statement": "11 May 2016 - 17 May 2016",
                        "Paid Status": "Not paid",
```

```
   “Order no:”: "123445666666",
 “Order Item no:”: "1666666",
“Shipment Type:”: "Dropshipping",
 “Shipment provider:”: "DEX",
                    "Reference": "1340",
                    "Comment": "",
                    "PaymentRefId": ""
                },
                {
                    "Transaction Date": "17 May 2016",
                    "Transaction Type": "Payment Fee",
                    "Transaction Number": "SG103EF-1P9VK1A",
"Daraz SKU": "Item test -123",
                    "Amount": "-0.62",
                    "VAT in Amount": "0.0672",
                    "WHT Amount": "0.0112",
                    "WHT included in Amount": "Yes",
                    "Statement": "11 May 2016 - 17 May 2016",
                    "Paid Status": "Not paid",
 “Order no:”: "123445666666",
 “Order Item no:”: "1666666",
“Shipment Type:”: "Dropshipping",
 “Shipment provider:”: "DEX",
                    "Reference": "1340",
                    "Comment": "",
                    "PaymentRefId": ""
                },
                {
                    "Transaction Date": "17 May 2016",
                    "Transaction Type": "Commission",
                    "Transaction Number": "SG103EF-1NVC2D6",
                    "Amount": "-2.86",
                    "VAT in Amount": "0.3400",
                    "WHT Amount": "0.2800",
                    "WHT included in Amount": "Yes",
                    "Statement": "11 May 2016 - 17 May 2016",
                    "Paid Status": "Not paid",
    “Order no:”: "123445666666",
 “Order Item no:”: "1666666",
“Shipment Type:”: "Dropshipping",
 “Shipment provider:”: "DEX",

                    "Reference": "1340",
                    "Comment": "",
                    "PaymentRefId": ""
                },
                {
                    "Transaction Date": "17 May 2016",
                    "Transaction Type": "Item Price Credit",
                    "Transaction Number": "SG103EF-1FF0JVW",
```

```
                    "Amount": "28.00",
                    "WHT included in Amount": "Yes",
                    "Statement": "11 May 2016 - 17 May 2016",
                    "Paid Status": "Not paid",
   “Order no:”: "123445666666",
  “Order Item no:”: "1666678",
 “Shipment Type:”: "Dropshipping",
  “Shipment provider:”: "DEX",
                    "Reference": "1340",
                    "Comment": "",
                    "PaymentRefId": ""
                }
            ]
        }
    }
}
```

```
}
```

GetSeller

Definition

Use this call to get seller information by the current user ID.

Request URI: https://api.sellercenter.daraz.pk?Action=GetSeller

Parameters

Field	Type	Description
Action	string	GetSeller Name of the API that is to be called. Mandatory.
Format	string	The response format, with XML as the default. Can be XML or JSON. Optional.
Timestamp	datetime	The current time in ISO8601 format (e.g., Timestamp=2016-04-01T10:00:00+02:00 for Berlin). Mandatory.
UserID	string	The ID of the user making the call. Mandatory.
Version	string	The API version against which this call is to be executed. The current version is "1.0". Mandatory.
Signature	string	The cryptographic signature, authenticating the request. You must create this value by computing the SHA256 hash of the request, using the API key of the user specified in the UserID parameter. Mandatory.

Result sample

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SuccessResponse>
  <Head>
    <RequestAction>GetSeller</RequestAction>
    <RequestId>0bb6885a15242052952821948eedd5</RequestId>
    <ResponseType>Sender</ResponseType>
    <Timestamp>2018-04-20T06:21:33+00:00</Timestamp>
  </Head>
  <Body>
    <SellerId>100039446</SellerId>
    <ShortCode>PKXXXX</ShortCode>
    <Email>sample@daraz.com</Email>
    <Name>Shop name</Name>
    <NameCompany>Company name</NameCompany>
    <Cb>false</Cb>
  </Body>
</SuccessResponse>
```

FAQ

Frequently Asked Questions

How to list product variations with the *CreateProduct* API?

The product variations must be listed with a single API request. That is, in the request body of the *CreateProduct* API, list the product variations in the tags. You can use the "AssociatedSku" tag if you want to add some SKUs to an existing product.

Why is the total number of products shown on the Seller Center portal different from that returned by the *GetProduct* API?

The total number of products shown on the Seller Center portal is the number of SKUs, but the number of products returned by the *GetProduct* A PI is the number of items. That is, SKUs include variations of items.

Please also note that the returned results by API calls are determined by the specified parameters like Filter, Limit, Offset, and time conditions. The maximum number of products returned by a single *GetProduct* call is 500. If no limit value is specified, the *GetProduct* call returns 20 products by default.

Why does the *GetDocument* API fail to return the needed documents?

Make sure that the status of the specified order item is “RTS” (by calling the *SetStatusToReadyToShip* API) before calling the *GetDocument* API. Otherwise, the error message E034 will be returned.

Why does the *GetBrands* API fail to return a known brand?

Check whether the specified parameters filter out some brands. The maximum number of brands that can be returned by a single request is 1,000. If the brand cannot be found in the Seller Center portal either, contact the service team.

What does the “liverejected” QC status (returned by the *GetQcStatus* API) mean?

The “liverejected” QC status means that online SKUs are made offline by the QC team because of product qualification reasons.