dealhub

Pricing API

Overview

The following document will describe how to use DealHub pricing API that calculates the price for an item or items.

Pricing API have two types of requests:

1. Products parameters - This query receives a list of SKUs and returns a list of parameters required per each SKU.

2. Products pricing - This query gets a list of SKUs, general parameters, and the parameters above and returns a price for each SKU in the request

---

Authorization:

The header of each request described below must include DealHub Authentication Token, which is generated by CPQ and shared with a consuming application in advance.

```
Authorization: Bearer 1ASnqsus6favhWHn.CYSpB8ehznjIzCcN
```

dealhub

Products Parameters Request

This service returns a list of parameters required to calculate the price for each SKU.

Request

Request type: POST

URL: */api/v1/products/parameters*

Query body format: JSON

Query body example:

```
{
  "currency": "GBP",
  "geo": "UK",
  "version": "version 1",
  "playbook": "playbook name",
  "skus": [
    "A-2342342",
    "B-2342342"
  ]
}
```

Query body explanation:

Name	Description	Mandatory
skus	The list of SKUs to evaluate	Yes
currency	The currency ISO code to evaluate the rules upon	Yes
version	The version name to retrieve the parameters for. If not provided within the request, the system will use the active version	No
playbook	The playbook name (not display name) in case there is more than one playbook	No
geo	The geography to be used for the pricing calculation	Yes

Response

The response returns a list of parameters required to calculate the price for each SKU.

Format: JSON

Response blocks:

- skus - The list of SKUs that DealHub evaluated
- for each SKU the response will have 2 different blocks:
  - `product_factors` - A list of product-specific factors required for price calculation of the given SKU
  - `parameters` - The list of parameters needed to resolve all of the relevant pricing formulas, considering the provided currency, regardless of source (exchange rate, formulas, etc.)

- Each element in above blocks will have the same format:
  - `name` - The name of parameter to return
  - `type` - The expected value type

JSON example:

```
{
  "skus": [
    {
      "sku": "A-2342342",
      "parameters_for": {
        "product_factors": [
          {
            "name": "duration",
            "type": "number"
          },
          {
            "name": "quantity",
            "type": "number"
          }
        ],
        "parameters": [
          {
            "name": "user.a-parameters",
            "type": "number"
          },
          {
            "name": "qg2.another-parameter",
            "type": "number"
          },
          {
            "name": "qg5.my-question",
            "type": "number"
          },
          {
            "name": "qg5.text-question",
            "type": "text"
          }
        ]
      }
    },
    {
      "sku": "B-2342342",
      "parameters_for": {
        "product_factors": [
          {
            "name": "quantity",
            "type": "number"
          },
          {
            "name": "duration",
            "type": "number"
          }
        ],
        "parameters": [
          {
            "name": "user.a-parameters",
            "type": "number"
          },
          {
            "name": "qg2.another-parameter",
            "type": "number"
          }
        ]
      }
    }
  ]
}
```

Error handling

HTTP code	When to return	Error Message
400	The requested version could not be found	“version <version name> could not be found”
400	Version was not provided and there is no active version	“Could not find an active version, please provide a specific”
400	If the currency provided is not in the list of the currencies defined in DealHub	“Currency ISO <ISO> is not supported in DealHub. Supported currencies: <list of defined currencies>”
400	Received playbook is not available	“playbook <playbook name> is not available in the requested version”
400	Playbook was not sent but there are more than one playbook available	“playbook name is required”
403	If the request is not authenticated	“Unauthenticated”

dealhub

Products Pricing Request

The service that returns a price for each SKU in the request.

Request

Request type: POST

URL: /api/v1/products/pricing

Query body format: JSON

Query body example:

```
"currency": "GBP",
 "geo": "UK",
 "version": "version 1",
 "playbook": "playbook name",
 "skus": [
   {
     "id" : "1",
     "sku": "A-2342342",
     "parameters": [
       {
         "name": "user.user_dur_hours",
         "value": 5
       },
       {
         "name": "qg3.counter",
         "value": 10
       },
       {
         "name": "user.a-parameters",
         "value": 1345
       },
       {
         "name": "qg2.another-parameter",
         "value": 19.4
       },
       {
         "name": "qg5.my-question",
         "value": 46
       },
       {
         "name": "qg5.text-question",
         "value": "Enterprise"
       },
       {
         "name": "general.geo",
         "value": "LATAM"
       },
       {
         "name": "general.currency",
         "value": "USD"
       },
       {
         "name": "qg6.question-name",
         "value": 247
       }
     ]
```

dealhub

```
      },
      {
        "id" : "2",
        "sku": "B-2342342",
        "parameters": [
          {
            "name": "user.user_dur_hours",
            "value": 24
          },
          {
            "name": "qg5.parameter3",
            "value": 46
          },
          {
            "name": "user.a-parameters",
            "value": 42
          },
          {
            "name": "qg2.another-parameter",
            "value": 6
          }
        ]
      }
    ]
  }
```

Query body explanation:

Name	Description	Mandatory
currency	The currency ISO code to base the pricing on. If not provided, the system will return all existing prices.	Yes
skus	The list of SKUs. Each element in the list should include: - **id** - a unique identifier that represents the ordinal position of this SKU in the request. - **sku** - SKU of the product. - **parameters** - List of parameters required for price calculation. Each element in the parameters list must have a name and a value.	Yes
geo	The geographic region to be used for the pricing calculation	Yes
version	The version name to retrieve the parameters for. If not provided, the system will use the active version.	No
playbook	The playbook name in case multiple playbooks exist.	No

dealhub

Response

The response will return the list of SKUs and their prices.

Format: JSON

Response blocks:

- `currency` - The currency in which the prices are shown
- `version` - The version for which the prices are related
- `playbook` - The playbook name for which the prices are related
- `skus` - a list of SKUs. for each SKU we will have the following fields:
    - `id` - the identifier that represents the ordinal position as sent in the request
    - `sku` - The product's SKU
    - `price` - the calculated price per unit in the requested currency (0 if there was an error)
    - `error` - an error message in case something went wrong

JSON example:

```
{
  "currency": "GBP",
  "version": "version 1",
  "playbook": "playbook name",
  "skus": [
    {
      "id": "1",
      "sku": "A-2342342",
      "price": 134.24,
      "error": ""
    },
    {
      "id": "2",
      "sku": "B-2342342",
      "price": 0
      "error": "Pricing could not be calculated"
    }
]
}
```

Error handling

HTTP code	When to return	Error Message
400	When one of the SKU is not available in the version	“sku: <sku> could not be found in version <version name>”
400	The requested version could not be found	“version <version name> could not be found”
400	If the currency provided is not in the list of the currencies defined in DealHub	“Currency ISO <ISO> is not supported in DealHub. Supported currencies: <list of defined currencies>”
400	Received playbook is not available	“playbook <playbook name> is not available in the requested version”
400	Playbook was not sent but there is more then one playbook available	“playbook name is required”
403	If the request is not authenticated	“Unauthenticated”

Additional considerations:

- If a price could not be calculated - error will contain the text: “Pricing could not be calculated”
- Product of type other than “standard price” - will have a price of 0 and an error of “Price type <price type> is not supported
- Only Fix price bundles supported by this API