API Usage Steps

To get started, follow these steps:

1. Create a My Mouser Account if you do not already have one
2. Generate a new API key
3. Review step 3 below, “API Explorer Details”, on how to use the API Explorer page
4. Review step 4 below, “Creating Requests”, on how to create API requests
5. Submit an order

Step 1 – Create a My Mouser Account

This is an optional step. If you already have a My Mouser account, you can skip this step. To create an account, go here and sign up for a My Mouser account.

Step 2 – Generate a new API key

After logging in, go to your My Mouser account page.

Under personal information a new link has been added to sign up for Mouser APIs.

Note: You may also want to add a billing and shipping address if you don’t already have them.

Follow the link for “APIs” and fill out the form to create a new API Key.

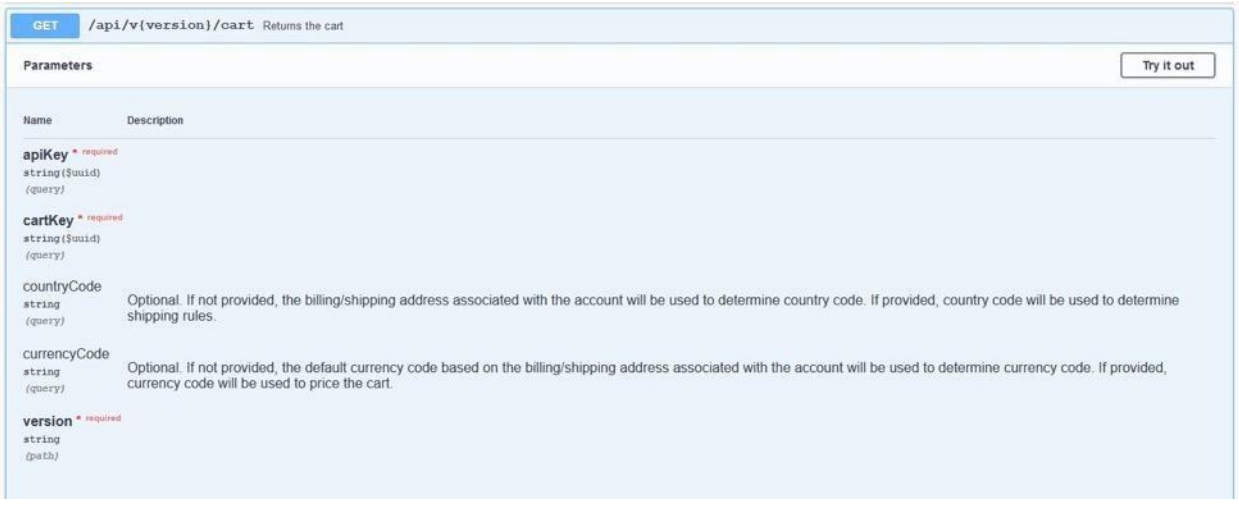
After filling out and submitting the form, your new API Key will show up for you to use with the Mouser Cart and Order APIs.

Step 3 – API Explorer Details

1. Go to the API Explorer page
2. *Note: You must have JavaScript enabled in your browser to use the API Explorer page.*

3. *Note: Only secure transactions, "https", are supported through Mouser's APIs.*

4. *Note: The API Explorer page is not the only way to test Mouser API's, you can also use other programs like Fiddler, Postman, or write your own client/server side code.*
5. *Note: The API Explorer page is not a sandbox. Any orders that are created will be placed against your account.*
6. The API Explorer page lists all available APIs, each API's endpoints and provides you with an easy to-use test area to make API requests.
7. API endpoints are broken up into categories or tags, like "Cart" and "Order". Each category is a separate API. Currently the only two supported APIs are Cart and Order. Categories describe the overall function of the endpoints associated with them. For example, Cart endpoints provide cart functionality like adding items to a cart or updating a cart.
8. Each endpoint has an HTTP method associated with it. Currently only "GET" and "POST" are supported.
9. Any POST endpoint will generally require a request body. The Options Query endpoint, "/api/v{version}/order/options/query", is one example where a request body is optional.
10. You can filter by tags or categories by entering in the name of the category to filter. For example, enter "Order" in the "Filter by tag" box. Notice only the "Order" level endpoints are shown.
11. From the API Explorer page, you can click on any of the listed endpoints to see more details about what is needed to create a request and see sample responses.
12. On the API Explorer page, click on the "/api/v{version}/cart" endpoint to expand the details. The first section listed is the request section.



13. Parameters are listed along the left-hand side. For example: "apiKey" and "cartKey". Any required parameters will be marked with "* required" text. If you don't see the required text, then the parameter is optional.

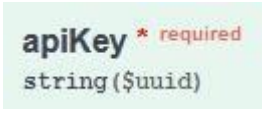
Data Types

Data types define what can be stored in the values of the properties. For example, with the property “MouserPartNumber”, the value it supports is string. A string is a sequence of characters that needs to be wrapped in quotes. So in JSON, the name/value pair would be: “MouserPartNumber”: “546-1590B”.

String

String can be any sequence of characters like “United States” and can be more specific like a unique identifier, “d8bf9628-0165-42b9-ab74-623e91051ab9”. All string values will need to be wrapped in double quotes in JSON.

Unique identifiers will show on the API Explorer page as ($uuid) as seen in the image below.



Number

Number types are decimal values, like 131.50. In JSON, they do not need to be stored in quotes. On the API Explorer page number types will show as number ($double).

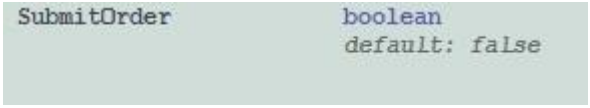
Integer

Integer types are numerical values like 15. In JSON, they do not need to be wrapped in quotes. On the API Explorer page integer types will show as integer ($int32)

Boolean

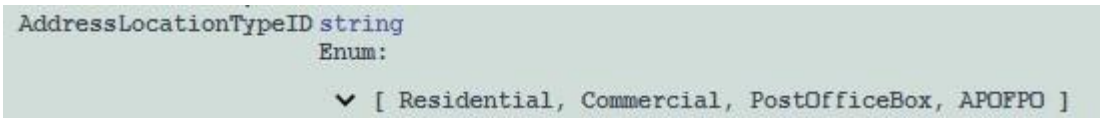
Boolean values can only be true or false. Example: “SubmitOrder”: true or “SubmitOrder”: false.

If a value is not given for a boolean type, it will default to false.



Enum

The Enum type is a string type, but it provides a list of values to choose from. See the sample image below. In this example, the property AddressLocationTypeID can be any of the listed values, like “Residential” or “Commercial”. Like any string values, JSON format requires it to be wrapped in double quotes.



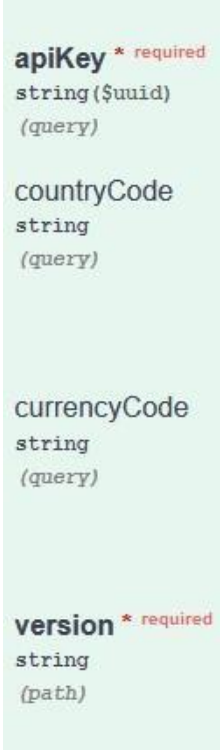
Null

The null type means there isn’t a value or the value is unknown.

Parameter Types

There are three different parameter types, **query**, **path**, and **request body**. Under each parameter, you will see the parameter type and expected value of the parameter.

Ex: See the image below. Notice the “apiKey” parameter has a type of string (unique identifier) and is a query parameter type.



Query Parameters

Our API Explorer page will build the request URL based on the query and path parameters. All you need to do is supply them in the given text boxes.

For example: under the "/api/v{version}/cart" endpoint, click the button "Try it out". Notice you now see text boxes for the query and path parameters like "apiKey" and "cartKey".

Path Parameters

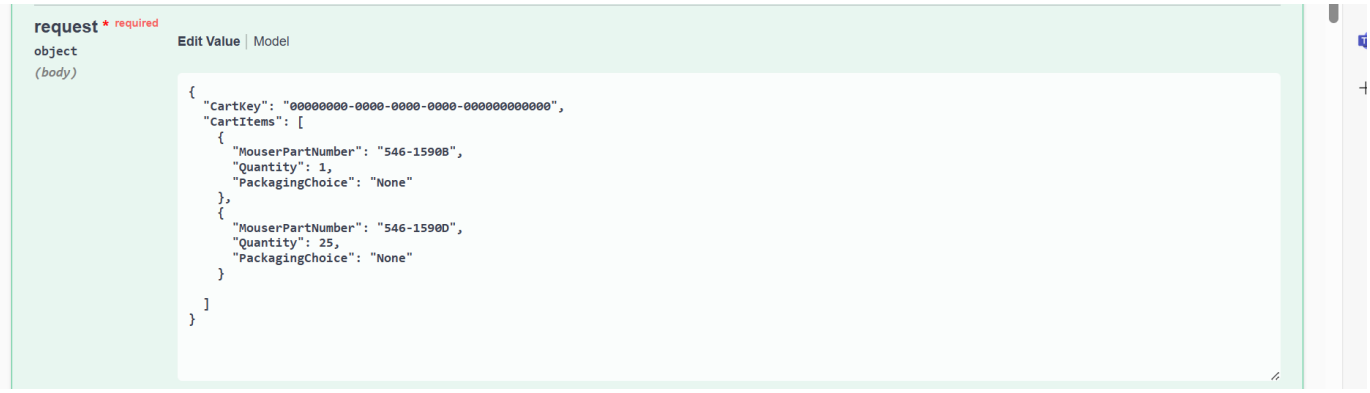
In the image above, the only path parameter is "version". Like query parameters, to fill in a path parameter on the API Explorer page, just fill in the value in the supplied text box.

Request Body Parameters – See the below image. Request body parameters will be marked with the (body) text and generally any POST method will require a request body to make the request.

The API Explorer page will give sample values for all request body parameters.

Ex: See the image below. A sample value is provided and in the case below the request body is shown in JSON. JSON is a way of storing data in an easy-to-understand format. JSON stores values in a collection of name/value pairs. As you see below, the "CartKey" would be the name or property and the part after the colon is the value. Any string data type would need to be wrapped in quotes. Example: "MouserPartNumber": "546-1590B". Numerical values do not need to be wrapped in quotes. If you aren't sure if the request body is valid JSON, go here, and paste in your request body to check if it's valid.

If any properties begin with a left bracket, "[", it means one or more values are supported. So, in the image above, the type "CartItems" can accept multiple objects (part numbers). Each additional part number would need to be separated by a comma. See the image below for a sample.



If you are not using the API Explorer page to test

Query parameters are in red in the request URL below. Query parameters consist of a key/value pair. Ex: Property name = Property value. The first query parameter will be followed by a question mark. All other query parameters will be followed by an ampersand.

Values in curly braces, {{apiKey}}, are place holder values and should be replaced with actual values.

https://api.mouser.com/api/v{{version}}/cart?apiKey={{apiKey}}&cartKey={{cartKey}}&countryCode={{countryCode}}&currencyCode={{currencyCode}}

Path parameters are included in the base URL of the request. Notice in the URL below, "v{{version}}" is a path parameter, and the curly braces should be replaced by actual values. So the base URL would look like "https://api.mouser.com/api/v1.0".

https://api.mouser.com/api/v{{version}}/cart?apiKey={{apiKey}}&cartKey={{cartKey}}&countryCode={{countryCode}}&currencyCode={{currencyCode}}

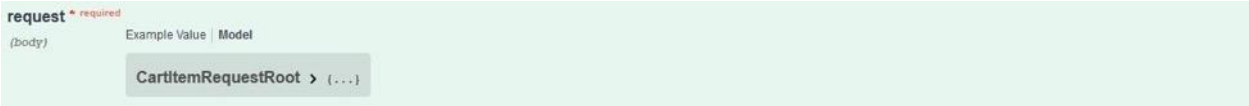
Request Body

If you are using another method of testing Mouser API's, it would still be beneficial to you to go to the API Explorer page and copy any of the sample request body's you are trying to test.

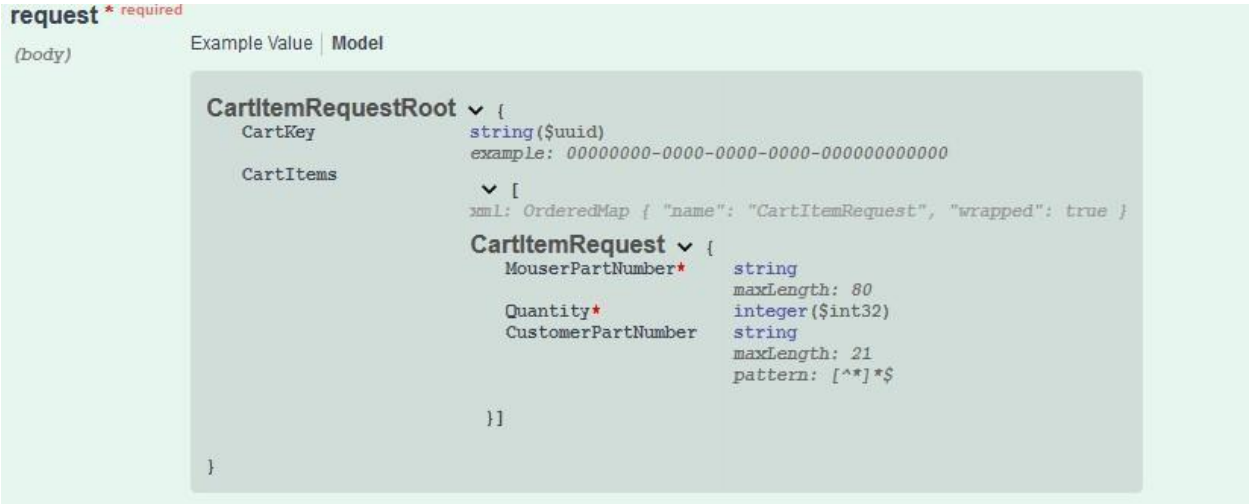
Request Body Samples

As mentioned earlier, each request that accepts a request body will have a sample request body or object model. The API Explorer page will let you drill into the object model to view the list of available properties, property types, max lengths, patterns, and if the property is required.

See the image below. To view the model, click on the “Model” text, next to the Example Value.



To drill into the object, click on the right arrow to expand the object’s properties.



Property level Info

Additional information about individual properties is listed to the right of the property name. See sample image below.

Max/Min Lengths

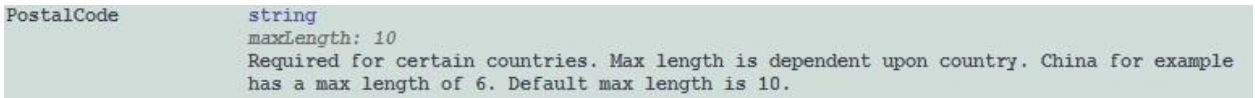
In the image above, “AddressOne” has a max length of 60 characters and minimum length of 3. Any values that fall outside of the range of 3 – 60 characters will be flagged as an error.

Required

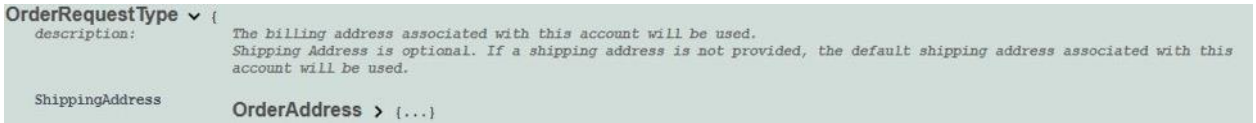
Any properties that are required to have a value will be marked with a red asterisk. In the image above CountryCode, FirstName and LastName are required.

Comments

Comments are just helpful information about properties. For example, in the image below notice PostalCode is not marked as required, but it is required for certain countries.



Comments can also be assigned at the object level as seen below.



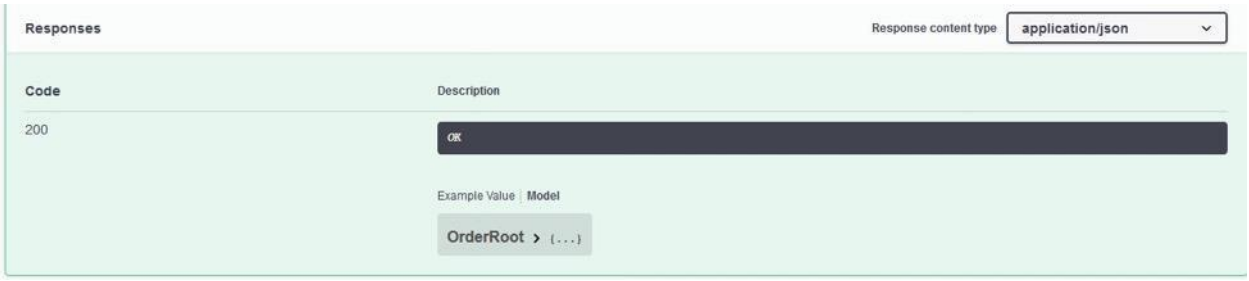
Patterns

Patterns are Regular Expressions that verify a particular value matches a pattern. For example, CustomerPartNumber cannot contain an asterisk character.

```
CustomerPartNumber    string
                      maxLength: 21
                      pattern: [^*]*$
```

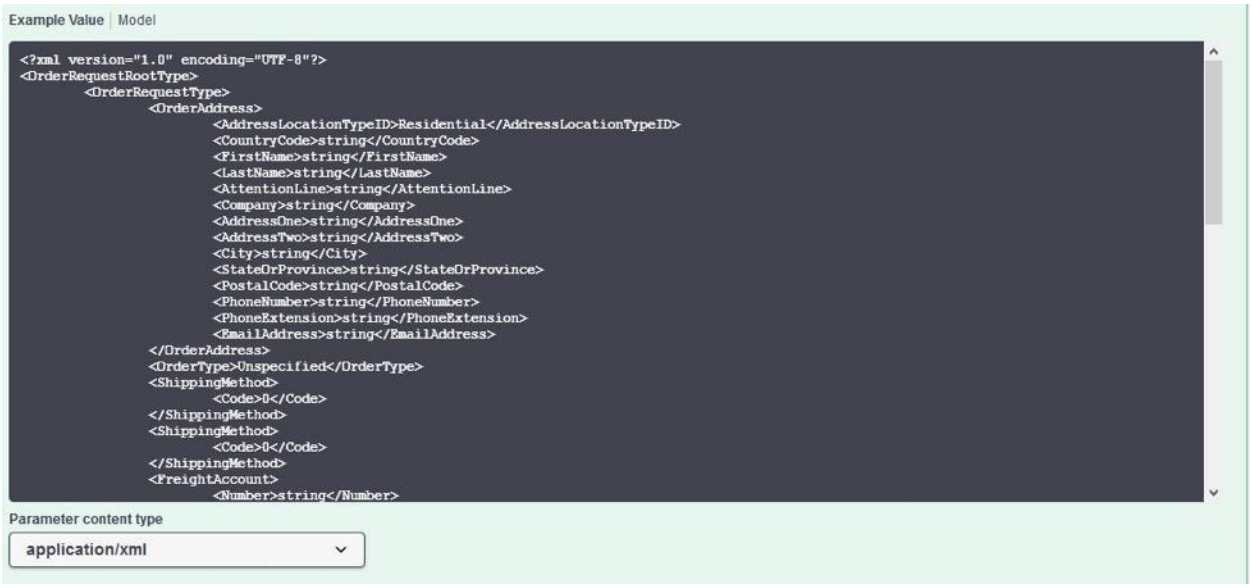
Response Object Samples

Each API request generated will receive a response. The API Explorer page will also provide sample responses for all API endpoints. In the API Explorer, scroll down to the section titled “Responses” as seen below. Like requests, responses also allows you to drill into the response object or model.



Request/Response Content Types

Mouser APIs also support other content types than JSON, like XML. On the API Explorer page, you can change the request content type to “application/xml” to see the sample object provided in XML as seen below.



Step 4 – Creating Requests

Now you are all ready to create requests. As mentioned earlier, request types are either GET or POST and POST types, typically require a request body.

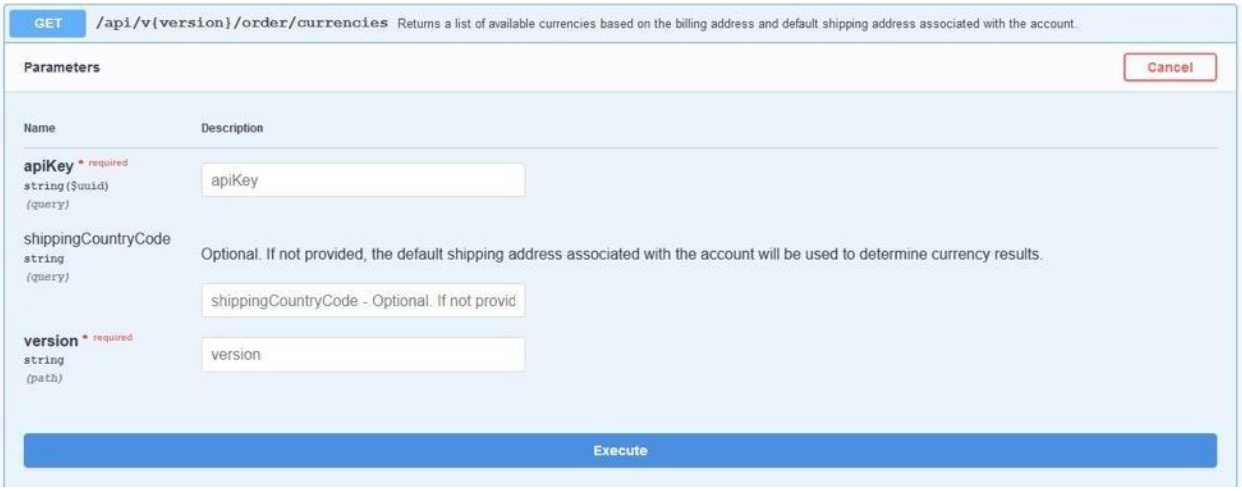
GET Requests

Let’s look at the “/api/v{version}/order/currencies” endpoint on the API Explorer page.

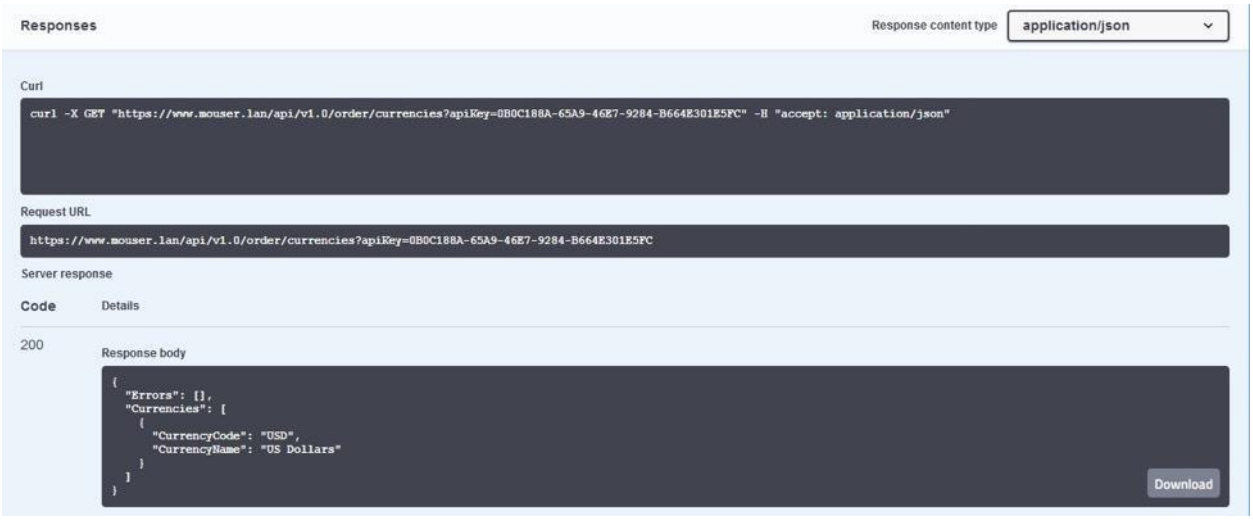
Notice it is a GET request, which means all parameters are either path or query parameters. Click on the endpoint to see the details. Once expanded, click on the “Try it out” button as seen below.

This will enable the request parameters to be editable as seen below. Fill in the required fields by suppling the API Key and version number in the text boxes.

Note: You can find your API Key here, just click the "Copy to Clipboard" button and paste it into the apiKey text box. Currently the only supported version is 1.0. So in the text box, enter 1.0.



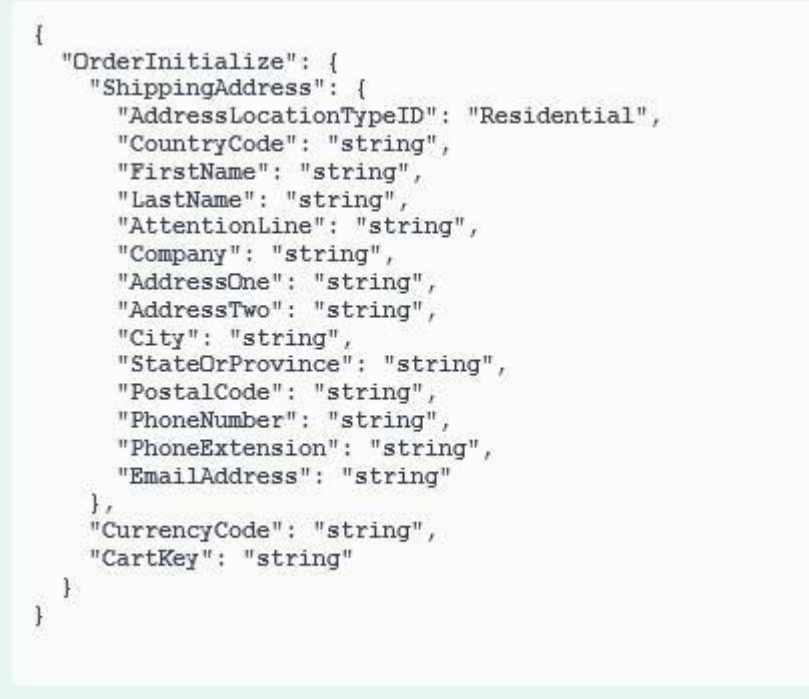
Now that the required fields are populated, click the "Execute" button to send the request. You should see a response similar to the one seen below.



POST Requests

Let's use the "/api/v{version}/order/options/query" endpoint as a POST example. Find the endpoint on the API Explorer page and click on it to expand the details, then click the "Try it out" button. Now the request body is editable as well as the request parameter text boxes. The "ShippingAddress" object is optional, but we will fill it out to use as an example. Looking at the sample image below, we see that each property will have a default value. CountryCode for example, has a default value of "string". This is just a dummy value and will need to be replaced with the actual value.

Note: All sample requests include default values, like "string", and if you don't remove the default value of "string" or replace it with your own, the value "string" will be processed as the actual value.



```
{
  "OrderInitialize": {
    "ShippingAddress": {
      "AddressLocationTypeID": "Residential",
      "CountryCode": "string",
      "FirstName": "string",
      "LastName": "string",
      "AttentionLine": "string",
      "Company": "string",
      "AddressOne": "string",
      "AddressTwo": "string",
      "City": "string",
      "StateOrProvince": "string",
      "PostalCode": "string",
      "PhoneNumber": "string",
      "PhoneExtension": "string",
      "EmailAddress": "string"
    },
    "CurrencyCode": "string",
    "CartKey": "string"
  }
}
```

Most of the properties for "ShippingAddress" are self-explanatory except the CountryCode and StateOrProvince properties. These two properties require a specific ISO Code value.

To get the values for CountryCode and StateOrProvince, we can execute the "/api/v{version}/order/countries" endpoint. This endpoint will return a list of countries, plus any states the country may have. For example, in the image below we see the response object returns values like CountryName, CountryCode, and a "States" array. An array is just a list of one or more values.

Response body


```
{
  "Countries": [
    {
      "CountryName": "United States",
      "CountryCode": "US",
      "States": [
        {
          "StateName": "Alabama",
          "StateCode": "AL"
        },
        {
          "StateName": "Alaska",
          "StateCode": "AK"
        },
        {
          "StateName": "American Samoa",
          "StateCode": "AS"
        },
        {
          "StateName": "Arizona",
          "StateCode": "AZ"
        },
        {
          "StateName": "Arkansas",
          "StateCode": "AR"
        },
```


To finish creating the request body for "/api/v{version}/order/options/query", we will use the values from the CountryCode and StateCode properties. In the image below, notice "string" has been replaced with "US" and "TX", signifying I want to use a ship to address with my country as the United States and a state of Texas. Finish replacing all of the dummy values with actual values.

The two other properties, “CurrencyCode” and “CartKey” are also optional, but if you needed to fill them out “CurrencyCode” would need to be populated with the “CurrencyCode” value returned in the “/api/v{version}/order/currencies” endpoint. See the example above under “GET Requests”.

Now that all values are populated, remember to fill in the apiKey and version properties, then click execute. You should see a request body similar to the sample image below.

Response body


```
    },
    "Shipping": {
      "Methods": [
        {
          "Method": "UPS Ground",
          "Rate": 8.89,
          "Code": 1
        },
        {
          "Method": "UPS 3rd Day",
          "Rate": 13.89,
          "Code": 36
        },
        {
          "Method": "UPS 2nd Day (End of Day)",
          "Rate": 14.89,
          "Code": 2
        },
        {
          "Method": "UPS Next Day SAVER (End of Day)",
          "Rate": 23.89,
          "Code": 71
        },
        {
          "Method": "UPS Next Day",
          "Rate": 30.89,
          "Code": 7
        },
```


Fault Tolerance

When you use our APIs, keep in mind that errors may be returned in the response body if there are issues with the request. Request errors will be returned in the response and will have a status code of 200. Any errors on our side will have a status code of 500.

Errors can be grouped under specific sections, like “ShippingAddress”, or at the root level of the response.

An example of a root level error is seen below. Notice the “Errors” element is not listed under a specific section or element like “ShippingAddress”. Instead it’s listed at the root of the Order element. In this case, the error means that the request body is not properly formatted JSON or XML.

In the sample image below, notice that the “ShippingAddress” section has the “Errors” array populated with an error. We only need to look at 3 properties in the Errors element, “Code”, “Message”, and “PropertyName”.

Code

The “Code” property is a short descriptor of the error type. In this case, the error type is required, meaning that we have a property in our ShippingAddress section that is not populated.

Please see the list below to see all possible values of the error code property:

- InternalError ○ The internal error code is a generic error meaning something went wrong (an error occurred) on the Mouser API side.
- EmptyCart ○ The empty cart error code means that at least one cart item is required.
- NoValidItemsInCart ○ This error code means that you have items in your cart but all of the cart items are in an error state.
- Invalid ○ The invalid error code is a generic error code that basically means the value that was sent with the request was invalid. For example, if a currency code is not supported in a particular country.
- Required ○ Required error codes specify that a particular value is required. Like the “FirstName” property mentioned above.
- MinLength ○ The minimum length error code means that the value provided does not meet the minimum number of characters. For example, if the minimum length for a property is 5 characters, and only 3 characters were provided, then a min length error will occur.
- MaxLength ○ The maximum length error code means the value provided exceeded the allowed amount of characters.
- InvalidFormat ○ Invalid format signifies the provided value is not in the correct format. For example, if a postal code was not in the correct format for the country.
- InvalidCharacters ○ Invalid characters are double-byte characters that are not supported in a specific country. Only certain countries support double-byte characters, like China. An example of a double-byte character is: ‘另’.
- NotAvailable ○ Not available means the supplied value is not available based on other values in your request. For example, if a shipping code was sent with the request that isn’t supported based on your billing/shipping address or cart items, then you will see this error code.

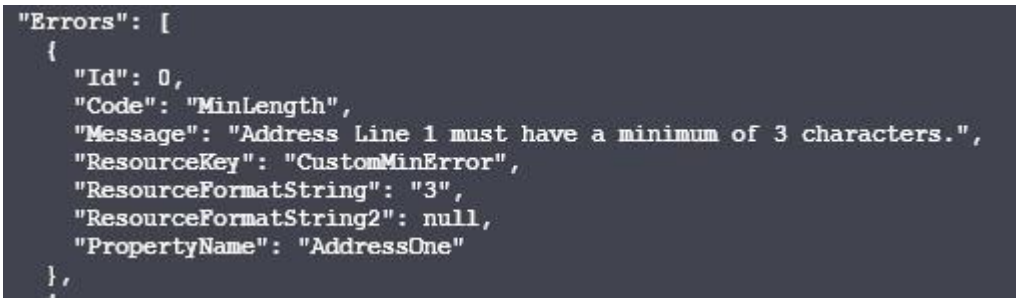
- NotAllowed ○ Not allowed means that you are trying to submit an order with items in you cart that are in an error state. Before submitting an order you must resolve those errored cart items or remove them from your cart.
- MinimumAmount

- Certain countries require a minimum order amount before submitting an order.

- NotFound ○ The not found error code would show up when you are trying to update or remove an item in your cart that does not exist. For example, if I want to update mouser part number: “546-1590B” and it doesn’t exist in my cart, I will see this error.
- NoScheduleLines
    - The no schedule lines error code would show up when you are trying to remove the scheduled deliveries for the cart items in your cart and none of the cart items have scheduled deliveries saved before.

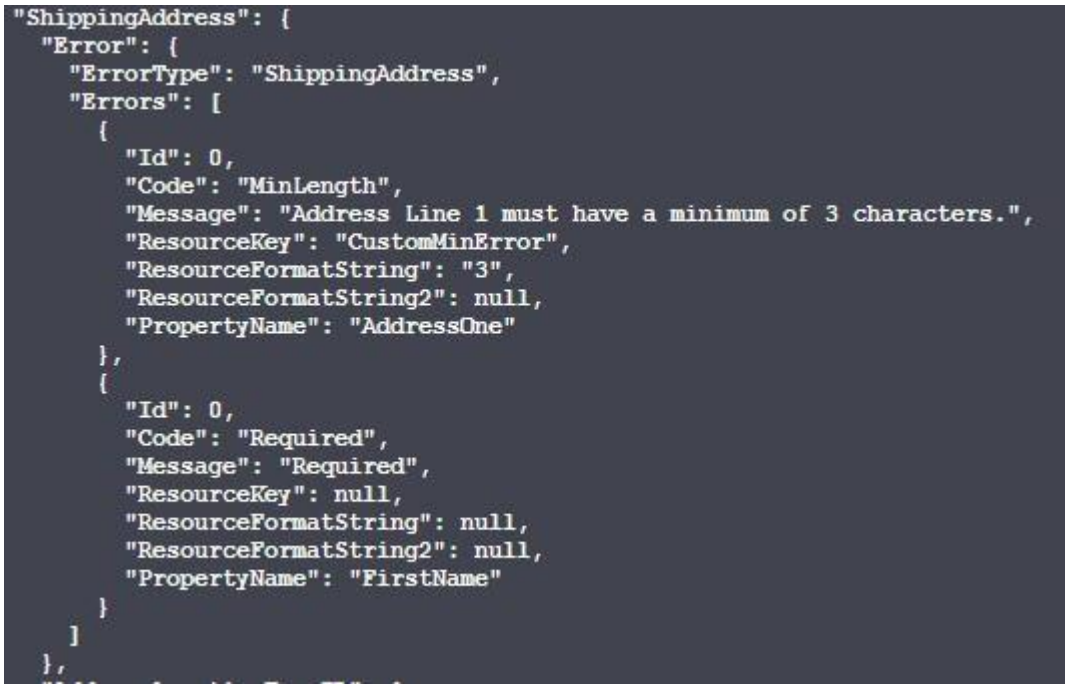
Message

The “Message” property is a full length message that explains what the error is. For example, in the image below, notice the message gives detailed information on how many characters are needed for the “AddressOne” property.



Property Name

The “PropertyName” property will reference a specific property in the request. In the sample image below notice that in the “ShippingAddress” object or section, we have a list of errors. This means that any errors listed in the errors section under “ShippingAddress” are all shipping address related errors. So when we see a property name like “AddressOne”, we know to review what data we sent in with our request in the “ShippingAddress” section and the “AddressOne” property.



Step 5 – Submit an Order

Before submitting an order, we must first build a cart. Creating a cart involves sending requests to Mouser's Cart API with the part number(s) you want to add to your cart. Keep in mind that **only Mouser part numbers** are supported.

Cart API

The Cart API functions very similarly to the cart experience you are familiar with on Mouser's website and the same validation/rules will apply. For example, minimum order quantity requirements and shipping restrictions will be enforced.

Note: Adding/updating a billing address is not supported through Mouser APIs. To add or update a billing address go here. Whenever a billing address is needed in our APIs, your billing address assigned in your My Mouser account will be used.

Note: A billing or shipping address is not required to create a cart. If they do not exist, the billing and shipping country will default to the United States.

To determine pricing for your cart, the default currency based on your billing and shipping address will be used. For example, if you have a billing and shipping address in the United States, the default currency is USD (US Dollars). You also have the ability to override the currency code we determine based

on your billing/shipping address. To override a currency code, just enter in a valid currency code in the text box on the API Explorer page as seen below.

Note: You must enter a valid currency for your country.

To determine cart based rules and shipping restrictions, the default country based on your billing and shipping address will be used. You may also override country code in the same manner as currency code.

Note: Only 100 cart items are allowed per request. And the total size of your cart cannot exceed 399 cart items.

The Cart API provides eight endpoints listed below.

1. **GET /api/v{version}/cart**
    a. Returns the cart
2. **POST /api/v{version}/cart**
    a. Updates the entire cart. If any cart items exist in the cart that were not sent with the request, they will be deleted.
    b. At least one part number is required
    c. If the quantity of the cart item sent in the request is not greater than zero, an error will be returned. The Cart API will not support updating a cart item quantity to zero.
3. **POST /api/v{version}/cart/items/insert**
    a. Adds one or more cart items to the cart.
    b. At least one part number is required
    c. If the cart key is not provided or empty, then a new cart key will be created.
4. **POST /api/v{version}/cart/items/update**
    a. Updates one or more cart items in the cart
    b. At least one part number is required
    c. If one or more cart items sent in do not match an existing cart item, an error will be returned.
    d. If the quantity of the cart item sent in the request is not greater than zero, an error will be returned.
5. **POST /api/v{version}/cart/item/remove**
    a. Removes a single cart item based on mouser part number
    b. At least one part number is required
    c. If one or more cart items sent in do not match an existing cart item in the cart, an error will be returned.
6. **POST /api/v{version}/cart/insert/schedule**
    a. Adds one or more scheduled deliveries to cart item(s).

b. The cart item(s) that needs to be scheduled should exist in the cart.

c. Schedule deliveries must pass scheduling restrictions, please see scheduling guidelines here.

Note: The endpoint insert/schedule should be used when there are no scheduled deliveries for any cart items. If you would like to update or add new deliveries for the cart items, use the update/schedule endpoint.

Request:

Below is the sample request that we can use to add scheduled releases for the cart items. In this example you will notice that we can add multiple scheduled deliveries to the cart item, which are separated by a comma. The property "ScheduleReleases" is a key value pair with Key as a date (format: MM/DD/YYYY) that you want your deliveries to be scheduled and value is the quantity of items you want to be delivered for that delivery.


```
{
  "CartKey": "96165de4-5e56-4806-8e73-611bafc8e6d2",
  "ScheduleCartItems": [
    {
      "MouserPartNumber": "5169-URB121000",
      "ScheduledReleases": [
        {
          "key": "10/10/2020",
          "value": 2
        },
        {
          "key": "10/29/2020",
          "value": 1
        }
      ]
    }
  ]
}
```

Response:

The above request fails in one of the restrictions “Requested delivery dates cannot fall on a weekend.”(see scheduling guidelines here). The below response shows the error ”Requested delivery date cannot fall on a weekend.” right above the schedule release line that fails the validation.


```
"PartsPerReel": 0,
"ScheduledReleases": [
  {
    "Errors": [
      {
        "Id": 0,
        "Code": "NotAllowed",
        "Message": "Requested delivery date cannot fall on a weekend.",
        "ResourceKey": "SchDateWeekend",
        "ResourceFormatString": null,
        "ResourceFormatString2": null,
        "PropertyName": "Key"
      }
    ],
    "ScheduleReleaseLine": {
      "Key": "10/10/2020",
      "Value": 1
    }
  },
  {
    "Errors": null,
    "ScheduleReleaseLine": {
      "Key": "10/29/2020",
      "Value": 1
    }
  }
],
```

A successful request will show the scheduled release lines without any error messages in the response as below.


```
{
  "Errors": [],
  "CartKey": "96165de4-5e56-4806-8e73-611bafc8e6d2",
  "CurrencyCode": "USD",
  "CartItems": [
    {
      "Errors": [],
      "MouserATS": 1,
      "Quantity": 5,
      "PartsPerReel": 0,
      "ScheduledReleases": [
        {
          "Errors": null,
          "ScheduleReleaseLine": {
            "Key": "10/14/2020",
            "Value": 1
          }
        },
        {
          "Errors": null,
          "ScheduleReleaseLine": {
            "Key": "10/29/2020",
            "Value": 1
          }
        }
      ],
      "InfoMessages": [
```


7. POST /api/v{version}/cart/update/schedule

a. Updates one or more scheduled deliveries for cart item(s).
b. If new scheduled deliveries are provided for the cart item, then a new scheduled delivery will be created.
c. The cart item(s) that needs to be scheduled should exist in the cart.

d. Schedule deliveries must pass scheduling restrictions, please see scheduling guidelines here.

Request:

Here is the sample request for updating the scheduled deliveries that we previously added using the /insert/schedule endpoint for the cart item "5169-URB121000". If you notice I have updated the quantity for the delivery with the date “10/14/2020”, from 1 to 2 and I have added a new scheduled delivery with date “12/01/2020” and quantity 1.


```
{
  "CartKey": "96165de4-5e56-4806-8e73-611bafc8e6d2",
  "ScheduleCartItems": [
    {
      "MouserPartNumber": "5169-URB121000",
      "ScheduledReleases": [
        {
          "key": "10/14/2020",
          "value": 2
        },
        {
          "key": "12/01/2020",
          "value": 1
        }
      ]
    }
  ]
}
```

Response:

Below is the response with updated scheduled deliveries, if you notice the quantity for the scheduled delivery dated “10/14/2020” is updated with quantity 2 and the new schedule delivery with date “12/01/2020” is added to the existing list of scheduled deliveries.

```
"Errors": [],
"MouserATS": 1,
"Quantity": 5,
"PartsPerReel": 0,
"ScheduledReleases": [
  {
    "Errors": null,
    "ScheduleReleaseLine": {
      "Key": "10/14/2020",
      "Value": 2
    }
  },
  {
    "Errors": null,
    "ScheduleReleaseLine": {
      "Key": "10/29/2020",
      "Value": 1
    }
  },
  {
    "Errors": null,
    "ScheduleReleaseLine": {
      "Key": "12/01/2020",
      "Value": 1
    }
  }
],
"InfoMessages": [
```

8. POST /api/v{version}/cart/deleteall/schedule

a. Removes all scheduled deliveries for cart item(s) based on the cart key.

Response:

After execution of this endpoint the scheduled deliveries for each cart item for the provided cart key will be deleted. The below response shows that the “ScheduleReleases” property has no value for its cart item.


```
{
  "Errors": [],
  "CartKey": "96165de4-5e56-4806-8e73-611bafc8e6d2",
  "CurrencyCode": "USD",
  "CartItems": [
    {
      "Errors": [],
      "MouserATS": 1,
      "Quantity": 5,
      "PartsPerReel": 0,
      "ScheduledReleases": null,
      "InfoMessages": [
        "Due to the size or weight of this product your shipping charge may have increased.",
        "Available in USA only."
      ],
      "MouserPartNumber": "5169-URB121000",
      "MfrPartNumber": "URB121000",
      "Description": "12V 100Ah LiFePO4 13.46 x 6.81 x 8.35\"",
      "CartItemCustPartNumber": "",
      "UnitPrice": 1422.3,
      "ExtendedPrice": 7111.5,
      "LifeCycle": "",
      "Manufacturer": "Ultralife",
      "SalesMultipleQty": 1,
      "SalesMinimumOrderQty": 1
    }
  ],
```


Cart API Sample Request:

Below is a sample request for **/api/v{version}/cart/items/insert**. In this example I want to create a new cart, so I have left CartKey empty which will create a new Cart and CartKey. In the response, you will find the new CartKey. Also notice that I’m adding multiple items to the cart, which are separated by a comma.


```
{
  "CartKey": "",
  "CartItems": [
    {
      "MouserPartNumber": "546-1590B",
      "Quantity": 1
    },
    {
      "MouserPartNumber": "546-1590D",
      "Quantity": 1
    }
  ]
}
```

Cart API Response

See the sample below for an example of what data is returned. *Note: any information returned in the "InfoMessages" element are not errors, it's just informational messages only.*


```
{
  "Errors": [],
  "CartKey": "96165de4-5e56-4806-8e73-611bafc8e6d2",
  "CurrencyCode": "USD",
  "CartItems": [
    {
      "Errors": [],
      "MouserATS": 1,
      "Quantity": 1,
      "PartsPerReel": 0,
      "ScheduledReleases": null,
      "InfoMessages": [
        "Due to the size or weight of this product your shipping charge may have increased.",
        "Available in USA only."
      ],
      "MouserPartNumber": "5169-URB121000",
      "MfrPartNumber": "URB121000",
      "Description": "12V 100Ah LiFePO4 13.46 x 6.81 x 8.35\"",
      "CartItemCustPartNumber": "",
      "UnitPrice": 1650,
      "ExtendedPrice": 1650,
      "LifeCycle": "",
      "Manufacturer": "Ultralife",
      "SalesMultipleQty": 1,
      "SalesMinimumOrderQty": 1
    }
```


Order API

The Order API gives you the ability to view order details for an order, get a list of supported countries, states, currencies, and most importantly to submit an order.

Like the Cart API, pricing and order rules are determined based on the billing address associated to your account and the default shipping address or the shipping address sent with the API request.

Note: Any orders created through Mouser's Order API are actual orders, not test orders.
The Order API provides five endpoints listed below.

1. **POST /api/v{version}/order/options/query**

The /options/query endpoint returns your billing and shipping address, currency code, a list of shipping methods plus any rates, payment methods, any freight accounts, any tax certificates, any VAT accounts, and any saved credit cards for your account.

A billing and shipping address is required to use the options/query endpoint.

Credit cards are only supported through the Order API if you have a saved credit card on your account. To manage saved cards, go here.

Request

The entire request body for options/query is optional. So if you don't want to override your shipping address or currency code, you can remove them from your request.

For example, the sample object model provided through the API Explorer page is seen in the below image.


```
{
  "OrderInitialize": {
    "ShippingAddress": {
      "AddressLocationTypeID": "Residential",
      "CountryCode": "string",
      "FirstName": "string",
      "LastName": "string",
      "AttentionLine": "string",
      "Company": "string",
      "AddressOne": "string",
      "AddressTwo": "string",
      "City": "string",
      "StateOrProvince": "string",
      "PostalCode": "string",
      "PhoneNumber": "string",
      "PhoneExtension": "string",
      "EmailAddress": "string"
    },
    "CurrencyCode": "string",
    "CartKey": "string"
  }
}
```


However, I just want to override the currency code, so I've removed the shipping address and cart key from the request as seen in the image below.


```
{
  "OrderInitialize": {
    "CurrencyCode": "EUR"
  }
}
```


Note: Currency code is optional. The default currency based on your billing/shipping address will be used if not provided.

Note: Shipping address is optional, if not sent, the default shipping address on the account is used.

Note: Cart Key is optional. However if a cart key is provided you will get accurate shipping rates and shipping methods returned.

Response

Note: If the response body for options/query shows 'null' for Shipping rates, then either you didn't send in a cart key or your cart is empty. Any shipping methods with "Bill Recipient" in the method name will not have shipping rates.

In the response you will see a list of available shipping/payment choices based on your billing and shipping address and cart if provided. You will need to use the numerical values of the shipping and payment codes when you submit an order.

For example, if I want to choose UPS Ground when I submit an order, I will use the value '1' from the "Code" property's value as seen below.


```
"Shipping": {
  "Methods": [
    {
      "Method": "UPS Ground",
      "Rate": 19.77,
      "Code": 1
    },
    {
      "Method": "UPS 3rd Day",
      "Rate": 39.6,
      "Code": 36
    },
    {
      "Method": "UPS 2nd Day (End of Day)",
      "Rate": 19.39,
      "Code": 2
    },
```


Freight Accounts

If you want to use your freight account, you need to choose a Shipping method that has Bill Recipient in the name. For example, "UPS Ground – Bill Recipient". Any freight accounts on your account will be returned in the Freight Accounts section as seen below. When you submit an order, you will need to send the value from the "Number" property or just send in a new freight account number. To manage your freight accounts, go here.

Note: When submitting an order, the freight account type does not need to be populated, only the freight number.


```
"FreightAccounts": [
  {
    "Number": "564829",
    "Type": "FedEx"
  },
  {
    "Number": "9876543",
    "Type": "UPS"
  },
```


Saved Cards

To use your saved credit cards when you submit an order, you will need to choose the payment method and send the last four digits of the credit card. For example, as seen in the image below, if I wanted to choose my saved Visa card, I would enter 31 and the last four digits of “4113” when I submit my order. To manage your saved cards, go here.


```
"Payment": {
  "PaymentTypes": [
    {
      "Method": 1,
      "Name": "Net 30",
      "CreditCardLastFourDigits": null
    },
    {
      "Method": 60,
      "Name": "Wire Transfer / Proforma",
      "CreditCardLastFourDigits": null
    },
    {
      "Method": 27,
      "Name": "COD",
      "CreditCardLastFourDigits": null
    },
    {
      "Method": 30,
      "Name": "Mastercard ending in 5114- Exp  01/19",
      "CreditCardLastFourDigits": "5114"
    },
    {
      "Method": 31,
      "Name": "Visa ending in 4113- Exp  01/19",
      "CreditCardLastFourDigits": "4113"
    }
  ],
```

2. **GET /api/v{version}/order/currencies**

Returns all available currencies based on your billing/shipping address. A billing and shipping address is required to use the /currencies endpoint.

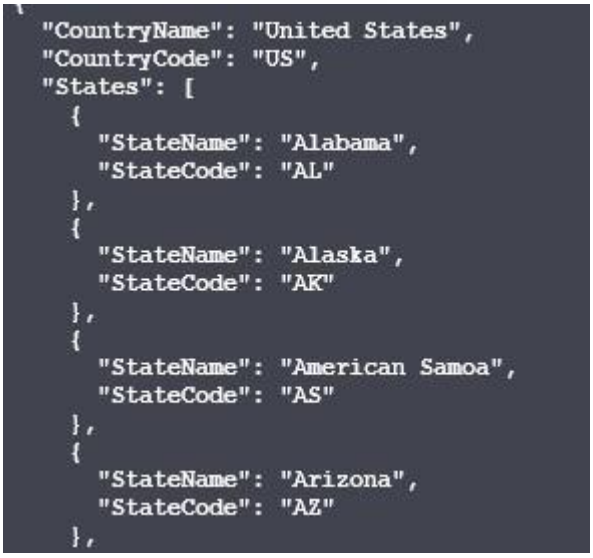
A shipping country code can be passed in to override the default shipping address on your account

To submit an order, you will need to send a currency code for the order. As seen in the image below, if I want to choose the currency US Dollars for my order, I would send the 3 digit ISO Code value of USD.

3. **GET /api/v{version}/order/countries**

Returns all available countries plus any state/provinces the country has. Allows user to filter countries by country code. Values returned from this endpoint can be used when you submit an order and you want to override the default shipping address on your account.

For example, if I wanted to ship to Arizona, I would send “AZ” as the state ISO Code and “US” as the country ISO code to submit an order.



4. POST /api/v{version}/order

Use this endpoint to submit a new order. As mentioned earlier, any orders created here will be placed against your account and a cart will need to be created before submitting an order.

If any errors exist in the billing address/shipping address/shipping method/payment/country or currency/scheduled releases, you will not be able to submit an order. If you're not sure you have an error, see the section on Fault Tolerance.

If any cart items are errored, you will need to either correct the error or remove the cart item before submitting an order.

An order number will be returned if a successful order is placed.

Note: Only Russian orders will not have an order number. Instead of an order number, you will see a message that the order was submitted.

Shipping address is optional, if not sent, the default shipping address on the account will be used.

If the cart contains backordered items, an order type (rush/complete) selection is required.

Order API Sample Request

Below are some sample requests for submitting an order, /api/v{version}/order.

Sample 1

In this example, I have chosen to ship my items “Rush”, which means ship immediately instead of waiting for all items to be in stock.

Note: “OrderType” is only required if your cart contains backordered items. If no value is provided, “OrderType” defaults to “Rush”.

My shipping choice is UPS Ground, which has a “Code” of 1. I’m using the saved Visa card on my account ending in 4113.

I’ve chosen to place my order in the USD currency and “SubmitOrder” is set to “false”. Which means I’m not ready to submit an order, I just want to review everything before I submit an order.

If you look at the sample request provided on the API Explorer page you will notice it is much larger and has many more elements and properties. But all of those elements/properties are not needed in every request. So I’ve stripped away what I don’t need from the request as seen below.


```
{
  "Order": {
    "OrderType": "Rush",
    "PrimaryShipping": {
      "Code": 1
    },
    "Payment": {
      "Method": 31,
      "CreditCardLastFourDigits": 4113
    },
    "CurrencyCode": "USD",
    "CartKey": "6f2efed2-d251-45f5-9ef8-9fc5f411e976",
    "SubmitOrder": false
  }
}
```


Sample 2

In this example, I’ve chosen to override the default shipping address on my account.

```
{
  "Order": {
    "ShippingAddress": {
      "AddressLocationTypeID": "Residential",
      "CountryCode": "DE",
      "FirstName": "John",
      "LastName": "Doe",
      "AddressOne": "100 Royal Ln.",
      "City": "Seattle",
      "StateOrProvince": "WA",
      "PostalCode": "98101",
      "PhoneNumber": "123-456-7890"
    },
    "OrderType": "Rush",
    "PrimaryShipping": {
      "Code": 1
    },
    "Payment": {
      "Method": 31,
      "CreditCardLastFourDigits": 4113
    },
    "CurrencyCode": "USD",
    "CartKey": "6f2efed2-d251-45f5-9ef8-9fc5f411e976",
    "SubmitOrder": false
  }
}
```

Sample 3

In this sample, my default shipping address is Germany (not shown), and I've chosen to enter my VAT Account number. Also notice that I've chosen to submit a new order and so "SubmitOrder" is set to true.

Note: Account numbers are returned from the /api/v{version}/order/options/query. To manage your VAT accounts go here.

Order API Sample Request – GST Business / Standard Invoicing for India

To use **GST Business Invoice** we need to have an account with Shipping Address and Billing Address both set to India as country code (**CountryCode** = "IN"), and it must have a "**VatAccountNumber**" present in the Request body. The "**CurrencyCode**" should be "INR", and it would throw an error message In the Response body if the "**CurrencyCode**" is USD.

The Shipping Address/ Billing Address should be a commercial address (i.e) it should have "**CompanyName**" field to avail a GST Business Invoice.

The shipping choice should be Mouser Select Shipping, which has a Primary Shipping "Code" of 99. Also, the only allowed payment methods for a GST Business Invoice would be NET Terms and Wire Transfer

- NET Terms - Payment "Method" - 1
- Wire Transfer – Payment "Method" – 60

*Note***:**

1. To bill outside of India do not include a GSTIN with your order.

2. To purchase without a company name do not include a GSTIN with your order.

3. To purchase using USD do not include a GSTIN with your order

4. To purchase using credit cards do not include a GSTIN with your order.

5. To ship using DHL, UPS, or FedEx do not include a GSTIN with your order.

6. IEC code is not required for a GST Invoice order.

Sample (GST Business Invoice):

```
{
  "Order": {
    "PrimaryShipping": {
      "Code": 99
    },

    "Payment": {
      "Method": "1",
    "TaxStatus": "Taxable",
          "VatAccountNumber" : "29AAECH3221K1ZL"
    },
    "currencyCode": "INR",
    "CartKey": "dcdf507b-313c-4b61-ad4f-fee7e9da02c7",
    "LanguageCode": "en-GB",
    "SubmitOrder": true
  }
}
```

To place a Standard Invoice Order "**VatAccountNumber**" should be empty or removed. Currency could be either INR or USD, Credit card would be the only mode of payment if currency is INR. IEC Code would be mandatory field.

Note:

- To purchase using NET Terms or Wire Transfer in INR include a GSTIN ("VatAccountNumber").
- To ship using Mouser Select Shipping include a GSTIN with your order.

Sample (Standard Invoice):

```
{
  "Order": {
    "PrimaryShipping": {
      "Code": 65
    },

    "IECCode": "AANFC0255C",
    "Payment": {
      "Method": "1",
    },
    "currencyCode": "INR",
    "CartKey": "dcdf507b-313c-4b61-ad4f-fee7e9da02c7",
    "LanguageCode": "en-GB",
    "SubmitOrder": true
  }
}
```

5. GET /api/v{version}/order/{orderNumber}

Use this endpoint to view the details for an order you created. To use the query, you must provide the order number, which is returned in the response body for the endpoint used to create an order, /api/v{version}/order.

Frequently Asked Questions

1. Why am I getting an unauthorized error when using Mouser APIs?
    a. There may be a few reasons you are receiving an unauthorized message. First, ensure that the API Key is correct. You can view your API Key and copy it to your clipboard by going here,
    b. It may be possible that one or more of your API accounts have been suspended due to API usage violations.
2. Why can't I submit an order?
    a. Here are a few things to consider:
        i. Your cart may be empty
        ii. Your cart may have one or more parts in an errored state. If so, please correct the errors or remove the part(s) from your cart.

iii. You may have other errors that are preventing you from submitting an order. Look at the response body and check for any errors that would be found under the “Errors” element.

iv. Make sure you set the “SubmitOrder” property value to true.

3. How do I create a cart key?
    a. Cart keys are created automatically when you insert items to your cart. Just leave the property “CartKey” empty and a new Cart Key will be created for you. Your new Cart Key will be found in the response body. *Note: remember that when you are adding more items to your cart after a cart is created to include your “CartKey” value. If the Cart Key is left empty, a new cart will be created with each request.*
4. What do I populate the version text box with on the API Explorer page?
    a. Currently the only supported version is 1.0

5. Why do I keep getting the error “Request data is missing or malformed”?
    a. Your request body is not properly formatted. If you are using JSON, please make sure the JSON is valid by going here, or any other JSON validator. If you are using XML, go here to validate.
6. Why do I keep getting the error “No match found For {part number}”?
    a. One reason you may receive this error is because you are not providing a mouser part number. The Cart API only supports Mouser part numbers.
    b. Check your part number for typos. You can also search on www.mouser.com for the part number and see what results are returned.
7. Why do I see the error “Authorization has been denied for this request?”
    a. Make sure you are entering the correct API Key from your API account page.
    b. If you are using the API Explorer page to test, check the text box for any spaces and remove them.
    c. Your account(s) may have been disabled due to API violations.