**GemGuide Pricing Direct API**

This documentation is intended for developers learning how to use the GemGuide Pricing Direct API.

**Setup**

If you are reading this documentation then you presumably already have an API key and have been registered as an authorized API user for the GemGuide Pricing Direct API. If do not already have an API key, please contact GemGuide support for further assistance. The API key will be required for all requests and you will need to provide the Username for your active Gemguide Subscription to Gemworld for call tracking.

The user must have a row in the **API\_Users** table in the Database, this should include their **name** and **api\_key** at minimum. Once this is added, Gemworld will set up the API key to be included in the **API\_TOKEN\_WHITELIST**. Once these steps are completed, the user can access the pricing API by providing only the **api\_key** header for the API request.

**Authentication**

The GemGuide Pricing Direct API uses a modified api\_key header ONLY so there is no authentication required.

Required Headers:

* **api\_key**: PROVIDED\_API\_KEY

**Request Structure**

The basic structure for requests is:

https://app.gemguide.com/prices-api/{route}?arg1=foo&arg2=bar

Where {route} is either gem or diamond depending on what pricing data you are after. All query args should be percent URL encoded.

For the Colored Gemstone route, an example request might look like:

https://app.gemguide.com/prices-api/gem?&name=Almandine%20Garnet&weight=1

For the Diamond route, an example request might look like:

https://app.gemguide.com/prices-api/diamond?name=Emerald&weight=1&color=G&clarity=IF/FL

**Authorization:**

In order for any of the above requests to work, you will need to provide the follow parameters as headers:

1. api\_key: This is the API key that GemGuide provided to you.

**Error Handling:**

Error responses use the following format:

{

code: 'error\_code',

message: 'This is a message describing the error in human friendly language'

}

Invalid requests may receive one of the following error responses:

* user\_unauthenticated: This means your API key is invalid or no longer registered.
* user\_client\_unauthenticated: This means the unique ID of the user (as detailed in the **Authentication** section above) is invalid or has since changed.
* user\_client\_expired: This means that it has been 30 days since the unique ID of the user has been updated. **A user is required to log in and be re-authenticated at least once every 30 days in order to prevent this error from happening.**

**Routes**

**Colored Gemstones**

The structure for the Colored Gemstone route is as follows:

https://app.gemguide.com/prices-api/gem?&name={name}&weight={weight}

There are two required arguments:

1. name: This argument is the name of the gem. It is **case sensitive** and must correspond **precisely** to the colored gemstone's name in the [GemGuide App](https://app.gemguide.com/). Space characters should be escaped as %20 i.e. Almandine%20Garnet
2. weight: This is the weight of the gemstone in carats.

The response will be an object of arrays like so:

{

"1": [ 2 ],

"2": [ 2 ],

"3": [ 3 ],

"4": [ 4 ],

"5": [ 5 ],

"6": [ 7 ],

"7": [ 9 ],

"8": [ 10 ],

"9": [ 12 ],

"10": [ 15 ]

}

The values 1 through 10 represent increasing levels of clarity (which typically implies a corresponding increase in price). Price values are returned as arrays since some values represent a range of prices from low to high (e.g. [9, 12]) rather than as a single, discrete price point. **Developers can check the size of the returned array to determine whether the value repesents a single price or a price range; one value means a single price, two values means a range of prices.**

**Error Handling:**

Error responses use the following format:

{

code: 'error\_code',

message: 'This is a message describing the error in human friendly language'

}

Invalid requests may receive one of the following error responses:

* no\_shape\_name\_provided: The request was missing a value for name
* no\_weight\_provided: The request was missing a value for weight
* invalid\_gem: The value for name does not match any gemstone names in the GemGuide database.
* invalid\_weight\_nan: The weight specified is not a number
* invalid\_weight: The weight specified falls outside the range of weights for the requested gem. The gem's valid weight range will be specified in the message property in the error response.
* server\_error: There was an unexpected server issue.

**Diamonds**

The structure for the Diamond route is as follows:

https://app.gemguide.com/prices-api/diamond?name={name}&weight={weight}&color={color}&clarity={clarity}

There are four required arguments:

1. name: This is the name of the diamond shape. It is **case sensitive** and must correspond **precisely** to the diamond shape's name in the [GemGuide App](https://app.gemguide.com/). Space characters should be escaped as %20 i.e. Old%20European
2. weight: This is the weight of the diamond in carats. **Exception: Baguettes require length in (mm) size instead of carat weight to access pricing.**
3. color: This is the **case sensitive** color value of the diamond. Colors values are specified as upper-case letters from D-M (i.e, D, E, F, G, H, I, J, K, L, M)
4. clarity: This is the **case sensitive** clarity value of the diamond. Clarity values must be one of the following: IF/FL, VVS1, VVS2, VS1, VS2, SI1, SI2, I1, I2, I3

The response will be an array of three arrays, each with three values, like so:

[

[

"-",

6930,

6230

],

[

"-",

5530,

5050

],

[

"-",

4700,

4200

]

]

The response contains a multi-dimensional array, a "table" of prices, based on the configuration specified in the query arguments. The top-level array values represent rows in the table, whereas each nested array represents columns within that row. The example API response from above could be represented as:

| - | 6930 | 6230 |

|-----|------|------|

| - | 5530 | 5050 |

|-----|------|------|

| - | 4700 | 4200 |

Note that the response is "centered" on the requested value, meaning the API always returns the previous and subsequent rows of data, as well as the preceeding and following values within the same row.

In programmatic terms, this means the exact price derived based on the query arguments is found at array[1][1]. The value at array[1][0] represents the same configuration except one clarity level higher and the value at array[1][2] represents the configuration at one clarify level lower. The values found in the preceeding (i.e. array[0]) and following (i.e. array[2]) rows representing the same clarity as array[1], but with the next lower or higher color value respectively.

Another way to visualize a given response could be like so (not precisely based on the above example):

[

D => [

IF/FL,

VVS1,

VVS2

],

E => [

IF/FL,

VVS1,

VVS2

],

F => [

IF/FL,

VVS1,

VVS2

]

]

In some instances, such as in the first diamond response example, there are no valid values for a specified configuration. For example, if you select IF/FL as your clarity level, there is no higher clarity level to reference. In these cases, instead of a price, the value will be filled in with a hyphen (-). This can also happen with colors, as you cannot go higher than D or lower than M.

To further elaborate this, if you set a color to be D (the highest color value) and clarity to be I3 (the lowest clarity value) you would get a response that looks like this:

[

[

"-",

"-",

"-"

],

[

1700,

1020,

"-"

],

[

1615,

935,

"-"

]

]

So in the above case, you can only reference lower color levels and higher clarity levels.

**Error Handling:**

Error responses use the following format:

{

code: 'error\_code',

message: 'This is a message describing the error in human friendly language'

}

Invalid requests may receive one of the following error responses:

* no\_shape\_name\_provided: The request was missing a value for name
* no\_color\_provided: The request was missing a value for color
* no\_clarity\_provided: The request was missing a value for clarity
* no\_weight\_provided: The request was missing a value for weight
* invalid\_shape: The name specified did not match a diamond shape name.
* invalid\_weight\_nan: The weight specified is not a number
* invalid\_weight: The weight specified falls outside the range of weights for the requested diamond. The diamond shape's valid weight range will be specified in the message property in the error response.
* invalid\_color: The color value did not match a valid color. Valid colors are specified at the top of this section (**Routes#Diamonds**)
* invalid\_clarity: The clarity value did not match a valid clarity. Valid clarities are specified at the top of this section (**Routes#Diamonds**)
* server\_error: There was an unexpected server issue.