

Discovery Document

 developers.google.com/discovery/v1/reference/apis

```
{
  "kind": "discovery#restDescription",
  "discoveryVersion": "v1",
  "id": string,
  "name": string,
  "canonicalName": string,
  "version": string,
  "revision": string,
  "title": string,
  "description": string,
  "icons": {
    "x16": string,
    "x32": string
  },
  "documentationLink": string,
  "labels": [
    string
  ],
  "protocol": "rest",
  "baseUrl": string,
  "basePath": string,
  "rootUrl": string,
  "servicePath": string,
  "batchPath": "batch",
  "endpoints": [
    {
      "endpointUrl": string,
      "location": string,
      "deprecated": boolean,
      "description": string
    }
  ],
  "parameters": {
    (key): {
      "id": string,
      "type": string,
      "$ref": string,
      "description": string,
      "default": string,
      "required": boolean,
      "format": string,
      "pattern": string,
      "minimum": string,
      "maximum": string,
      "enum": [
        string
      ],
      "enumDescriptions": [
        string
      ],
      "repeated": boolean,
      "location": string,
      "properties": {
```

```
      (key): (JsonSchema)
    },
    "additionalProperties": (JsonSchema),
    "items": (JsonSchema),
    "annotations": {
      "required": [
        string
      ]
    }
  }
},
"auth": {
  "oauth2": {
    "scopes": {
      (key): {
        "description": string
      }
    }
  }
},
"features": [
  string
],
"schemas": {
  (key): {
    "id": string,
    "type": string,
    "$ref": string,
    "description": string,
    "default": string,
    "required": boolean,
    "deprecated": boolean,
    "format": string,
    "pattern": string,
    "minimum": string,
    "maximum": string,
    "enum": [
      string
    ],
    "enumDescriptions": [
      string
    ],
    "enumDeprecated": [
      boolean
    ],
    "repeated": boolean,
    "location": string,
    "properties": {
      (key): (JsonSchema)
    },
    "additionalProperties": (JsonSchema),
    "items": (JsonSchema),
    "annotations": {
      "required": [
```

```

        string
    ]
}
},
"methods": {
    (key): {
        "id": string,
        "path": string,
        "httpMethod": string,
        "description": string,
        "deprecated": boolean,
        "parameters": {
            (key): {
                "id": string,
                "type": string,
                "$ref": string,
                "description": string,
                "default": string,
                "required": boolean,
                "deprecated": boolean,
                "format": string,
                "pattern": string,
                "minimum": string,
                "maximum": string,
                "enum": [
                    string
                ],
                "enumDescriptions": [
                    string
                ],
                "enumDeprecated": [
                    boolean
                ],
                "repeated": boolean,
                "location": string,
                "properties": {
                    (key): (JsonSchema)
                },
                "additionalProperties": (JsonSchema),
                "items": (JsonSchema),
                "annotations": {
                    "required": [
                        string
                    ]
                }
            }
        },
    },
    "parameterOrder": [
        string
    ],
    "request": {
        "$ref": string
    }
}

```

```
    },
    "response": {
      "$ref": string
    },
    "scopes": [
      (value)
    ],
    "supportsMediaDownload": boolean,
    "supportsMediaUpload": boolean,
    "mediaUpload": {
      "accept": [
        string
      ],
      "maxSize": string,
      "protocols": {
        "simple": {
          "multipart": true,
          "path": string
        },
        "resumable": {
          "multipart": true,
          "path": string
        }
      }
    },
    "supportsSubscription": boolean
  },
  "resources": {
    (key): {
      "methods": {
        (key): {
          "id": string,
          "path": string,
          "httpMethod": string,
          "description": string,
          "deprecated": boolean,
          "parameters": {
            (key): {
              "id": string,
              "type": string,
              "$ref": string,
              "description": string,
              "default": string,
              "required": boolean,
              "deprecated": boolean,
              "format": string,
              "pattern": string,
              "minimum": string,
              "maximum": string,
              "enum": [
                string
              ],
            },
          ],
        },
      },
    },
  },
}
```

```
    "enumDescriptions": [  
      string  
    ],  
    "enumDeprecated": [  
      boolean  
    ],  
    "repeated": boolean,  
    "location": string,  
    "properties": {  
      (key): (JsonSchema)  
    },  
    "additionalProperties": (JsonSchema),  
    "items": (JsonSchema),  
    "annotations": {  
      "required": [  
        string  
      ]  
    }  
  }  
},  
"parameterOrder": [  
  string  
],  
"request": {  
  "$ref": string  
},  
"response": {  
  "$ref": string  
},  
"scopes": [  
  (value)  
],  
"supportsMediaDownload": boolean,  
"supportsMediaUpload": boolean,  
"mediaUpload": {  
  "accept": [  
    string  
  ],  
  "maxSize": string,  
  "protocols": {  
    "simple": {  
      "multipart": true,  
      "path": string  
    },  
    "resumable": {  
      "multipart": true,  
      "path": string  
    }  
  }  
}  
},  
"supportsSubscription": boolean  
}  
},  
"deprecated": boolean,
```

```
    "resources": {
      (key): (RestResource)
    }
  }
}
```

Property Name	Value	Description	Notes
kind	string	The kind for this response. The fixed string <code>discovery#restDescription</code> .	
discoveryVersion	string	Indicate the version of the Discovery API used to generate this doc.	
id	string	The ID of the Discovery document for the API. For example, <code>urlshortener:v1</code> .	
name	string	The name of the API. For example, <code>urlshortener</code> .	
canonicalName	string	The canonical name of the API. For example, <code>Url Shortener</code> .	
version	string	The version of the API. For example, <code>v1</code> .	
revision	string	The revision of the API.	
title	string	The title of the API. For example, "Google Url Shortener API".	
description	string	The description of this API.	
icons	object	Links to 16x16 and 32x32 icons representing the API.	
icons.x16	string	The URL of the 16x16 icon.	
icons.x32	string	The URL of the 32x32 icon.	
documentationLink	string	A link to human-readable documentation for the API.	
labels[]	list	Labels for the status of this API. Valid values include <code>limited_availability</code> or <code>deprecated</code> .	

Property Name	Value	Description	Notes
<code>protocol</code>	<code>string</code>	The protocol described by the document. For example, REST.	
<code>rootUrl</code>	<code>string</code>	The root url under which all API services live.	
<code>endpoints[]</code>	<code>list</code>	A list of location-based endpoint objects for this API. Each object contains the endpoint URL, location, description and deprecation status.	
<code>endpoints[].endpointUrl</code>	<code>string</code>	The URL of the endpoint target host.	
<code>endpoints[].location</code>	<code>string</code>	The location of the endpoint.	
<code>endpoints[].description</code>	<code>string</code>	A string describing the host designated by the URL.	
<code>endpoints[].deprecated</code>	<code>boolean</code>	Whether this endpoint is deprecated.	
<code>parameters</code>	<code>object</code>	Common parameters that apply across all apis.	
<code>parameters.(key)</code>	<code>nested object</code>	Description of a single parameter.	
<code>parameters.(key).id</code>	<code>string</code>	Unique identifier for this schema.	
<code>parameters.(key).type</code>	<code>string</code>	The value type for this schema. A list of values can be found at the "type" section in the JSON Schema.	
<code>parameters.(key).\$ref</code>	<code>string</code>	A reference to another schema. The value of this property is the ID of another schema.	
<code>parameters.(key).description</code>	<code>string</code>	A description of this object.	
<code>parameters.(key).default</code>	<code>string</code>	The default value of this property (if one exists).	
<code>parameters.(key).required</code>	<code>boolean</code>	Whether the parameter is required.	

Property Name	Value	Description	Notes
<code>parameters.(key).format</code>	string	An additional regular expression or key that helps constrain the value. For more details see the Type and Format Summary.	
<code>parameters.(key).pattern</code>	string	The regular expression this parameter must conform to.	
<code>parameters.(key).minimum</code>	string	The minimum value of this parameter.	
<code>parameters.(key).maximum</code>	string	The maximum value of this parameter.	
<code>parameters.(key).enum[]</code>	list	Values this parameter may take (if it is an enum).	
<code>parameters.(key).enumDescriptions[]</code>	list	The descriptions for the enums. Each position maps to the corresponding value in the enum array.	
<code>parameters.(key).repeated</code>	boolean	Whether this parameter may appear multiple times.	
<code>parameters.(key).location</code>	string	Whether this parameter goes in the query or the path for REST requests.	
<code>parameters.(key).properties</code>	object	If this is a schema for an object, list the schema for each property of this object.	
<code>parameters.(key).properties.(key)</code>	nested object	A single property of this object. The value is itself a JSON Schema object describing this property.	
<code>parameters.(key).additionalProperties</code>	nested object	If this is a schema for an object, this property is the schema for any additional properties with dynamic keys on this object.	
<code>parameters.(key).items</code>	nested object	If this is a schema for an array, this property is the schema for each element in the array.	
<code>parameters.(key).annotations</code>	object	Additional information about this property.	

Property Name	Value	Description	Notes
<code>parameters.(key).annotations.required[]</code>	<code>list</code>	A list of methods that require this property on requests.	
<code>auth</code>	<code>object</code>	Authentication information.	
<code>auth.oauth2</code>	<code>object</code>	OAuth 2.0 authentication information.	
<code>auth.oauth2.scopes</code>	<code>object</code>	Available OAuth 2.0 scopes.	
<code>auth.oauth2.scopes.(key)</code>	<code>object</code>	The scope value.	
<code>auth.oauth2.scopes.(key).description</code>	<code>string</code>	Description of scope.	
<code>features[]</code>	<code>list</code>	A list of supported features for this API.	
<code>schemas</code>	<code>object</code>	The schemas for this API.	
<code>schemas.(key)</code>	<code>nested object</code>	An individual schema description.	
<code>schemas.(key).id</code>	<code>string</code>	Unique identifier for this schema. Example: <code>URL</code>	
<code>schemas.(key).type</code>	<code>string</code>	The value type for this schema. A list of values can be found at the "type" section in the JSON Schema.	
<code>schemas.(key).\$ref</code>	<code>string</code>	A reference to another schema. The value of this property is the ID of another schema.	
<code>schemas.(key).description</code>	<code>string</code>	A description of this object.	
<code>schemas.(key).default</code>	<code>string</code>	The default value of this property (if one exists).	
<code>schemas.(key).required</code>	<code>boolean</code>	Whether the parameter is required.	
<code>schemas.(key).deprecated</code>	<code>boolean</code>	Whether this schema is deprecated.	

Property Name	Value	Description	Notes
<code>schemas.(key).format</code>	<code>string</code>	An additional regular expression or key that helps constrain the value. For more details see the Type and Format Summary.	
<code>schemas.(key).pattern</code>	<code>string</code>	The regular expression this parameter must conform to.	
<code>schemas.(key).minimum</code>	<code>string</code>	The minimum value of this parameter.	
<code>schemas.(key).maximum</code>	<code>string</code>	The maximum value of this parameter.	
<code>schemas.(key).enum[]</code>	<code>list</code>	Values this parameter may take (if it is an enum).	
<code>schemas.(key).enumDescriptions[]</code>	<code>list</code>	The descriptions for the enums. Each position maps to the corresponding value in the <code>enum</code> array.	
<code>schemas.(key).enumDeprecated[]</code>	<code>list</code>	The deprecation status for the enums. Each position maps to the corresponding value in the <code>enum</code> array.	
<code>schemas.(key).repeated</code>	<code>boolean</code>	Whether this parameter may appear multiple times.	
<code>schemas.(key).location</code>	<code>string</code>	Whether this parameter goes in the query or the path for REST requests.	
<code>schemas.(key).properties</code>	<code>object</code>	If this is a schema for an object, list the schema for each property of this object.	
<code>schemas.(key).properties.(key)</code>	<code>nested object</code>	A single property of this object. The value is itself a JSON Schema object describing this property.	
<code>schemas.(key).additionalProperties</code>	<code>nested object</code>	If this is a schema for an object, this property is the schema for any additional properties with dynamic keys on this object.	

Property Name	Value	Description	Notes
<code>schemas.(key).items</code>	<code>nested object</code>	If this is a schema for an array, this property is the schema for each element in the array.	
<code>schemas.(key).annotations</code>	<code>object</code>	Additional information about this property.	
<code>schemas.(key).annotations.required[]</code>	<code>list</code>	A list of methods that require this property on requests.	
<code>methods</code>	<code>object</code>	API-level methods for this API.	
<code>methods.(key)</code>	<code>nested object</code>	An individual method description.	
<code>methods.(key).id</code>	<code>string</code>	A unique ID for this method. This property can be used to match methods between different versions of Discovery.	
<code>methods.(key).description</code>	<code>string</code>	Description of this method.	
<code>methods.(key).deprecated</code>	<code>boolean</code>	Whether this method is deprecated.	
<code>methods.(key).parameters</code>	<code>object</code>	Details for all parameters in this method.	
<code>methods.(key).parameters.(key)</code>	<code>nested object</code>	Details for a single parameter in this method.	
<code>methods.(key).parameters.(key).id</code>	<code>string</code>	Unique identifier for this schema.	
<code>methods.(key).parameters.(key).type</code>	<code>string</code>	The value type for this schema. A list of values can be found at the "type" section in the JSON Schema.	
<code>methods.(key).parameters.(key).\$ref</code>	<code>string</code>	A reference to another schema. The value of this property is the ID of another schema.	

Property Name	Value	Description	Notes
<code>methods.(key).parameters.(key).description</code>	string	A description of this object.	
<code>methods.(key).parameters.(key).default</code>	string	The default value of this property (if one exists).	
<code>methods.(key).parameters.(key).required</code>	boolean	Whether the parameter is required.	
<code>methods.(key).parameters.(key).deprecated</code>	boolean	Whether the parameter is deprecated.	
<code>methods.(key).parameters.(key).format</code>	string	An additional regular expression or key that helps constrain the value. For more details see the Type and Format Summary.	
<code>methods.(key).parameters.(key).pattern</code>	string	The regular expression this parameter must conform to.	
<code>methods.(key).parameters.(key).minimum</code>	string	The minimum value of this parameter.	
<code>methods.(key).parameters.(key).maximum</code>	string	The maximum value of this parameter.	
<code>methods.(key).parameters.(key).enum[]</code>	list	Values this parameter may take (if it is an enum).	
<code>methods.(key).parameters.(key).enumDescriptions[]</code>	list	The descriptions for the enums. Each position maps to the corresponding value in the <code>enum</code> array.	
<code>methods.(key).parameters.(key).enumDeprecated[]</code>	list	The deprecation status for the enums. Each position maps to the corresponding value in the <code>enum</code> array.	
<code>methods.(key).parameters.(key).repeated</code>	boolean	Whether this parameter may appear multiple times.	

Property Name	Value	Description	Notes
<code>methods.(key).parameters.(key).location</code>	<code>string</code>	Whether this parameter goes in the query or the path for REST requests.	
<code>methods.(key).parameters.(key).properties</code>	<code>object</code>	If this is a schema for an object, list the schema for each property of this object.	
<code>methods.(key).parameters.(key).properties.(key)</code>	<code>nested object</code>	A single property of this object. The value is itself a JSON Schema object describing this property.	
<code>methods.(key).parameters.(key).additionalProperties</code>	<code>nested object</code>	If this is a schema for an object, this property is the schema for any additional properties with dynamic keys on this object.	
<code>methods.(key).parameters.(key).items</code>	<code>nested object</code>	If this is a schema for an array, this property is the schema for each element in the array.	
<code>methods.(key).parameters.(key).annotations</code>	<code>object</code>	Additional information about this property.	
<code>methods.(key).parameters.(key).annotations.required[]</code>	<code>list</code>	A list of methods for which this property is required on requests.	
<code>methods.(key).parameterOrder[]</code>	<code>list</code>	Ordered list of required parameters. This serves as a hint to clients on how to structure their method signatures. The array is ordered such that the most significant parameter appears first.	
<code>methods.(key).scopes[]</code>	<code>list</code>	OAuth 2.0 scopes applicable to this method.	
<code>methods.(key).supportsMediaDownload</code>	<code>boolean</code>	Whether this method supports media downloads.	
<code>methods.(key).supportsMediaUpload</code>	<code>boolean</code>	Whether this method supports media uploads.	

Property Name	Value	Description	Notes
<code>methods.(key).mediaUpload</code>	<code>object</code>	Media upload parameters.	
<code>methods.(key).mediaUpload.accept[]</code>	<code>list</code>	MIME Media Ranges for acceptable media uploads to this method.	
<code>methods.(key).mediaUpload.maxSize</code>	<code>string</code>	Maximum size of a media upload, such as "1MB", "2GB" or "3TB".	
<code>methods.(key).supportsSubscription</code>	<code>boolean</code>	Whether this method supports subscriptions.	
<code>baseUrl</code>	<code>string</code>	[DEPRECATED] The base URL for REST requests.	
<code>basePath</code>	<code>string</code>	[DEPRECATED] The base path for REST requests.	
<code>servicePath</code>	<code>string</code>	The base path for all REST requests.	
<code>batchPath</code>	<code>string</code>	The path for REST batch requests.	
<code>methods.(key).path</code>	<code>string</code>	The URI path of this REST method. Should be used in conjunction with the <code>servicePath</code> property at the API-level.	
<code>methods.(key).httpMethod</code>	<code>string</code>	HTTP method used by this method.	
<code>methods.(key).request</code>	<code>object</code>	The schema for the request.	
<code>methods.(key).request.\$ref</code>	<code>string</code>	Schema ID for the request schema.	
<code>methods.(key).request.parameterName</code>	<code>string</code>	[DEPRECATED] Some APIs have this field for backward-compatibility reasons. It can be safely ignored.	
<code>methods.(key).response</code>	<code>object</code>	The schema for the response.	
<code>methods.(key).response.\$ref</code>	<code>string</code>	Schema ID for the response schema.	
<code>methods.(key).mediaUpload.protocols</code>	<code>object</code>	Supported upload protocols.	

Property Name	Value	Description	Notes
<code>methods.(key).mediaUpload.protocols.simple</code>	<code>object</code>	Supports uploading as a single HTTP request.	
<code>methods.(key).mediaUpload.protocols.simple.multipart</code>	<code>boolean</code>	True if this endpoint supports upload multipart media.	
<code>methods.(key).mediaUpload.protocols.simple.path</code>	<code>string</code>	The URI path to be used for upload. Should be used in conjunction with the <code>rootURL</code> property at the api-level.	
<code>methods.(key).mediaUpload.protocols.resumable</code>	<code>object</code>	Supports the Resumable Media Upload protocol.	
<code>methods.(key).mediaUpload.protocols.resumable.multipart</code>	<code>boolean</code>	<code>true</code> if this endpoint supports uploading multipart media.	
<code>methods.(key).mediaUpload.protocols.resumable.path</code>	<code>string</code>	The URI path to be used for upload. Should be used in conjunction with the <code>rootURL</code> property at the API-level.	
<code>resources</code>	<code>object</code>	The resources in this API.	
<code>resources.(key)</code>	<code>nested object</code>	An individual resource description. Contains methods and sub-resources related to this resource.	
<code>resources.(key).methods</code>	<code>object</code>	Methods on this resource.	
<code>resources.(key).methods.(key)</code>	<code>nested object</code>	Description for any methods on this resource.	
<code>resources.(key).methods.(key).id</code>	<code>string</code>	A unique ID for this method. This property can be used to match methods between different versions of Discovery.	
<code>resources.(key).methods.(key).path</code>	<code>string</code>	The URI path of this REST method. Should be used in conjunction with the <code>servicePath</code> property at the API-level.	

Property Name	Value	Description	Notes
<code>resources.(key).methods.(key).flatPath</code>	string	The URI path of this REST method in (RFC 6570) format without level 2 features ({+var}). Supplementary to the <code>path</code> property.	
<code>resources.(key).methods.(key).httpMethod</code>	string	HTTP method used by this method.	
<code>resources.(key).methods.(key).description</code>	string	Description of this method.	
<code>resources.(key).methods.(key).deprecated</code>	boolean	Whether this method is deprecated.	
<code>resources.(key).methods.(key).parameters</code>	object	Details for all parameters in this method.	
<code>resources.(key).methods.(key).parameters.(key)</code>	nested object	Details for a single parameter in this method.	
<code>resources.(key).methods.(key).parameters.(key).id</code>	string	Unique identifier for this schema.	
<code>resources.(key).methods.(key).parameters.(key).type</code>	string	The value type for this schema. A list of values can be found at the "type" section in the JSON Schema.	
<code>resources.(key).methods.(key).parameters.(key).\$ref</code>	string	A reference to another schema. The value of this property is the "ID" of another schema.	
<code>resources.(key).methods.(key).parameters.(key).description</code>	string	A description of this object.	
<code>resources.(key).methods.(key).parameters.(key).default</code>	string	The default value of this property (if one exists).	
<code>resources.(key).methods.(key).parameters.(key).required</code>	boolean	Whether the parameter is required.	


Property Name	Value	Description	Notes
<code>resources.(key).methods.(key).parameters.(key).deprecated</code>	<code>boolean</code>	Whether the parameter is deprecated.	
<code>resources.(key).methods.(key).parameters.(key).format</code>	<code>string</code>	An additional regular expression or key that helps constrain the value. For more details see the Type and Format Summary.	
<code>resources.(key).methods.(key).parameters.(key).pattern</code>	<code>string</code>	The regular expression this parameter must conform to.	
<code>resources.(key).methods.(key).parameters.(key).minimum</code>	<code>string</code>	The minimum value of this parameter.	
<code>resources.(key).methods.(key).parameters.(key).maximum</code>	<code>string</code>	The maximum value of this parameter.	
<code>resources.(key).methods.(key).parameters.(key).enum[]</code>	<code>list</code>	Values this parameter may take (if it is an enum).	
<code>resources.(key).methods.(key).parameters.(key).enumDescriptions[]</code>	<code>list</code>	The descriptions for the enums. Each position maps to the corresponding value in the <code>enum</code> array.	
<code>resources.(key).methods.(key).parameters.(key).enumDeprecated[]</code>	<code>list</code>	The deprecation status for the enums. Each position maps to the corresponding value in the <code>enum</code> array.	
<code>resources.(key).methods.(key).parameters.(key).repeated</code>	<code>boolean</code>	Whether this parameter may appear multiple times.	
<code>resources.(key).methods.(key).parameters.(key).location</code>	<code>string</code>	Whether this parameter goes in the query or the path for REST requests.	

Property Name	Value	Description	Notes
<code>resources.(key).methods.(key).parameters.(key).properties</code>	<code>object</code>	If this is a schema for an object, list the schema for each property of this object.	
<code>resources.(key).methods.(key).parameters.(key).properties.(key)</code>	<code>nested object</code>	A single property of this object. The value is itself a JSON Schema object describing this property.	
<code>resources.(key).methods.(key).parameters.(key).additionalProperties</code>	<code>nested object</code>	If this is a schema for an object, this property is the schema for any additional properties with dynamic keys on this object.	
<code>resources.(key).methods.(key).parameters.(key).items</code>	<code>nested object</code>	If this is a schema for an array, this property is the schema for each element in the array.	
<code>resources.(key).methods.(key).parameters.(key).annotations</code>	<code>object</code>	Additional information about this property.	
<code>resources.(key).methods.(key).parameters.(key).annotations.required[]</code>	<code>list</code>	A list of methods that require this property on requests.	
<code>resources.(key).methods.(key).parameterOrder[]</code>	<code>list</code>	Ordered list of required parameters. This serves as a hint to clients on how to structure their method signatures. The array is ordered such that the most significant parameter appears first.	
<code>resources.(key).methods.(key).request</code>	<code>object</code>	The schema for the request.	
<code>resources.(key).methods.(key).request.\$ref</code>	<code>string</code>	Schema ID for the request schema.	
<code>resources.(key).methods.(key).response</code>	<code>object</code>	The schema for the response.	
<code>resources.(key).methods.(key).response.\$ref</code>	<code>string</code>	Schema ID for the response schema.	

Property Name	Value	Description	Notes
<code>resources.(key).methods.(key).scopes[]</code>	<code>list</code>	OAuth 2.0 scopes applicable to this method.	
<code>resources.(key).methods.(key).supportsMediaDownload</code>	<code>boolean</code>	Whether this method supports media downloads.	
<code>resources.(key).methods.(key).supportsMediaUpload</code>	<code>boolean</code>	Whether this method supports media uploads.	
<code>resources.(key).methods.(key).mediaUpload</code>	<code>object</code>	Media upload parameters.	
<code>resources.(key).methods.(key).mediaUpload.accept[]</code>	<code>list</code>	MIME Media Ranges for acceptable media uploads to this method.	
<code>resources.(key).methods.(key).mediaUpload.maxSize</code>	<code>string</code>	Maximum size of a media upload, such as "1MB", "2GB" or "3TB".	
<code>resources.(key).methods.(key).mediaUpload.protocols</code>	<code>object</code>	Supported upload protocols.	
<code>resources.(key).methods.(key).mediaUpload.protocols.simple</code>	<code>object</code>	Supports uploading as a single HTTP request.	
<code>resources.(key).methods.(key).mediaUpload.protocols.simple.multipart</code>	<code>boolean</code>	<code>true</code> if this endpoint supports upload multipart media.	
<code>resources.(key).methods.(key).mediaUpload.protocols.simple.path</code>	<code>string</code>	The URI path to be used for upload. Should be used in conjunction with the <code>rootURL</code> property at the API-level.	
<code>resources.(key).methods.(key).mediaUpload.protocols.resumable</code>	<code>object</code>	Supports the Resumable Media Upload protocol.	

Property Name	Value	Description	Notes
<code>resources.(key).methods.(key).mediaUpload.protocols.resumable.multipart</code>	boolean	<code>true</code> if this endpoint supports uploading multipart media.	
<code>resources.(key).methods.(key).mediaUpload.protocols.resumable.path</code>	string	The URI path to be used for upload. Should be used in conjunction with the <code>rootURL</code> property at the API-level.	
<code>resources.(key).methods.(key).supportsSubscription</code>	boolean	Whether this method supports subscriptions.	
<code>resources.(key).deprecated</code>	boolean	Whether this resource is deprecated.	
<code>resources.(key).resources</code>	object	Sub-resources on this resource.	
<code>resources.(key).resources.(key)</code>	nested object	Description for any sub-resources on this resource.	

Google Maps Solar API: Getting "404 Entity Not Found" for Specific Buildings, How to Check Available Areas?

 stackoverflow.com/questions/77007954/google-maps-solar-api-getting-404-entity-not-found-for-specific-buildings-ho

Sebastian Römer 4111 bronze badge

2



I've been trying to experiment with the new Google Maps Solar API to assess solar potential for various buildings. However, I've encountered a consistent issue where I receive a "404 Entity Not Found" error message for certain buildings I'm interested in. Strangely, my API access seems to be functional as I can successfully query buildings in larger cities. Is there a method to determine the geographical areas where the Solar API is applicable and which buildings are accessible? It's puzzling because I know my access is working, but the error is persistent for specific structures. Any insights or guidance on debugging this problem would be greatly appreciated.

I tried using the example in the docs, this lat / long is working. Buildings in bigger cities are working but none in my area.

google-mapsgoogle-solar-api

2

How hard is it to browse the documentation? – MrUpsideDown Aug 31 at 7:31

Add a comment

2 Answers

1



Check whether or not your building is in a HIGH or MEDIUM region (<https://developers.google.com/maps/documentation/solar/coverage?hl=en>). In your API request you can change the *requiredQuality* according to where your building is situated.


answered Sep 12 at 13:09



Tom Rigter

1111 bronze badge

1

According to that map, my building sits well within a MEDIUM coverage zone but I'm still getting 404 requested entity was not found when using MEDIUM or LOW requiredQuality. – lbowes Sep 24 at 21:37 

Add a comment

1



The Google Solar API response seems to be unfortunately lacking at the moment. As another commenter has already alluded to, your location needs to be in an area of HIGH or MEDIUM coverage in order to receive an API response. However, as you have already discovered, if your location is not in a coverage area the API response will be a very uninformative error. Specifically (at least in my case) the following error:

```
{
  "error": {
    "code": 403,
    "message": "Requests from referer \u003cempty\u003e are blocked.",
    "status": "PERMISSION_DENIED",
    "details": [
      {
        "@type": "type.googleapis.com/google.rpc.ErrorInfo",
        "reason": "API_KEY_HTTP_REFERRER_BLOCKED",
        "domain": "googleapis.com",
        "metadata": {
          "service": "solar.googleapis.com",
          "consumer": "projects/x"
        }
      }
    ]
  }
}
```

This is incredibly unhelpful (and misleading) for most people and it would be much more useful if the API returned an error code such as NOT_FOUND. Hopefully in the future Google will update the API response to something more appropriate.

As for your inquiry about a method to check whether or not a location has solar data available, your best bet at the moment is to just look for and handle the 404 error.

```
$.getJSON(url, function (data) {
  // building data has been returned
}).fail(function(response) {
  if (response.status == 404) {
    // no building data returned
  }
});
```

edited Sep 20 at 20:51

answered Sep 20 at 18:24



brassmookie

35922 silver badges1313 bronze badges

Add a comment

Your Answer

Sign up or log in



Sign up using Google



Sign up using Facebook



Sign up using Email and Password

Post as a guest

Required, but never shown

By clicking "Post Your Answer", you agree to our terms of service and acknowledge that you have read and understand our privacy policy and code of conduct.

Not the answer you're looking for? Browse other questions tagged [google-maps](#)[google-solar-api](#) or ask your own question.

How can I use the SolarAPI to obtain an image or TIFF file of a roof with solar panels?

 stackoverflow.com/questions/77236255/how-can-i-use-the-solarapi-to-obtain-an-image-or-tiff-file-of-a-roof-with-solar

1



In our application, we are currently working on energy planning for users' homes. To provide them with a basic plan tailored to their location, we require detailed user address information

I have already reviewed the documentation and identified several valuable data points we can obtain. From building insights, we can access information such as solar-panel count, yearly production, and other crucial details for the roof. Additionally, we have integrated data layers that provide monthly and hourly flux and shading information and images(.tiff file).

Then I have integrated data layers from which we can monthly/horly fulx and shades but for exmple below image Monthly flux image/tiff file

However, we've encountered two challenges. Firstly, I attempted to download and view .tiff files from the documentation but was unable to do so using regular image viewers. The documentation mentions this limitation, but as developers, we aim to display these images or overlay them on a map with GeoTIFF data.

enter image description here

Furthermore, the documentation states that GeoTIFF files cannot be directly used as overlay images with the Maps JavaScript API. How can we make these files viewable to our users within our React application?

However, our primary objective is to obtain detailed photos with solar panels installed on the roof. Is there a way to achieve this? The documentation only mentions access to data layers, but in their demo, they showcase images with solar panels that appear to be adjustable."

Doc:- solar-panel and also they are adujstable

"Finally, we're curious about obtaining detailed photos of roofs with solar panels, similar to what's demonstrated, and we greatly appreciate your assistance. Thanks in advance!"

asked 14 hours ago



rushil patel

1122 bronze badges

Add a comment


Solar API release notes

 developers.google.com/maps/documentation/solar/release-notes

- [Home](#)
- [Products](#)
- [Google Maps Platform](#)
- [Documentation](#)
- [Solar API](#)
- [Support](#)

Was this helpful?

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

Subscribe to these release notes. 

This page is updated with each new release of the Solar API. The changelog lists releases by date and includes any new features, bug fixes and significant performance improvements.

See the [Getting Started](#) documentation for information on how to get started using the Solar API.

August 08, 2023

Solar API

Announcing the General Availability (GA) release of the Solar API. See the [Release Notes](#) for information about this release and for all other releases.

If you are a new user, see [Set Up](#) in the Google Cloud Console to start the installation process.

2023-06-29

In July of 2023, Google will release a change to the format of URLs returned in the response to a call to a Data Layers request. To prepare for this release, write your apps to support both URL formats. By preparing for the new URL format now, your apps will not be affected by this update.

For more information, see [Data Layers request](#).

2023-05-09

Preview: Launched a **Restricted Preview** of the Solar API as part of Google Maps Platform on Google Cloud.

- Launched a restricted preview of the **new Solar API**, hosted as part of Google Maps Platform on Google Cloud. This new API will replace the now-deprecated EarthEngineSolar API, and, to minimize disruption, we encourage you to migrate as soon as feasible.
 - Revamped project-level access
 - To get access, please reach out to your contact on the Solar API team.
 - Access is manually granted on a per-project basis, and is independent from access to the EarthEngineSolar API.
 - Added new documentation
 - Set new terms of service
 - Integrated with Cloud Monitoring and Cloud Billing

Calls to Solar API endpoints will begin appearing on projects' Billing Dashboard (with a **100% discount** during the Preview).
 - Renamed endpoints and methods
 - `earthenginesolar.googleapis.com` -> `solar.googleapis.com`
 - `solarInfo:get` -> `dataLayers:get`
 - `buildings:findClosest` -> `buildingInsights:findClosest`
 - Configured new quota limits

Replaced per-day and per-quarter request quotas with per-minute quotas.

 - Quota limits are set at the project level and begin at 60 queries per minute for each endpoint.
 - Find a project's quota at <https://console.cloud.google.com/google/maps-apis/quotas>
- Deprecated the existing EarthEngineSolar API.

The deprecated EarthEngineSolar API will continue to work while the new Solar API is in Preview.
- Expanded medium-quality (`imagery_quality: 'MEDIUM'`) data coverage
 - Expanded medium-quality data availability for Germany, the UK, France, Spain, Italy, and Florida.
 - Expanded medium-quality financial data availability (via `BuildingInsights`) for a significantly larger portion of Florida.

2022-02-28

- Added different quality levels of data.
- Added quality level to returned data.
- Added ability to specify the required quality level.

2020-09-08

- Added dataset specifier for early access to non-production data.
- Added specification of panel parameters (width, height, name, place, capacity, lifetime).

- Added height of roof segments.
- Added information about building ground footprints.
- Added information about unsegmented roof size.
- Added panel-by-panel production values.

2019-06-21

- Added bounding boxes to building and roof segment structures.
- Added segment index to roof segment summaries, to unambiguously link each summary to more detailed information.
- Added documentation for new raster endpoint which will allow access to imagery and detailed solar potential information.

2018-07-20

Initial release.

Was this helpful?

Recommended for you

Set up your Google Cloud project

This guide shows how to set up your Google Cloud project before using the Google Maps Platform APIs. While you may have completed some of these steps in the Getting started with Google Maps Platform page, this topic provides additional, useful

Updated Oct 5, 2023

Solar API Concepts

The Solar API provides solar potential data through the `buildingInsights` and `dataLayers` endpoints. To use Solar API data, understanding the following concepts may be helpful: A building's solar potential is largely based on the amount of sunlight

Updated Oct 5, 2023

Solar API coverage

The Solar API provides solar data for hundreds of millions of buildings across the world. Imagery quality can be HIGH, MEDIUM, or LOW. The Google Maps Platform team is constantly working to improve coverage for our API services. The map below shows

Updated Oct 5, 2023

Frequently Asked Questions

 developers.google.com/maps/documentation/solar/faq

- [Home](#)
- [Products](#)
- [Google Maps Platform](#)
- [Documentation](#)
- [Solar API](#)
- [Support](#)

Was this helpful?

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

Help

How do I migrate to the new Solar API from the previous version?

Review our migration guide.

Where can I get help?

Check out Support Options for Solar API.

What do these error codes mean?

Read more about HTTP status codes.

Project configuration

What permissions do I need on my Cloud project?

See the documentation.

Can I use my API key with more than one Cloud project?

No, only use your API key for one Cloud project. If you have more Cloud projects, each should have their own API key. Note that, within a single project, it's normal to have several API keys.

Cost

How do I limit costs?

See the Budgeting documentation for strategies and tips.

Solar model

How often are data refreshed?

New data are added all the time, but we don't have a specific refresh rate for specific regions. Some regions use imagery that is several years old. Please check the dates of the relevant imagery, and feel free to file feedback.

Where can I learn more about the assumptions for the solar financial analysis?

Google uses information about government solar incentives, along with other financial data and models, from Clean Power Research.

Are chimneys and skylights taken into consideration by the API when making an assessment of solar panel placement?

Yes, with height maps chimneys and skylights are identified as small rectangles, and solar panels are excluded by the model.

Can the API see inside of buildings?

No, just the roofs.

Can the API make assessments for a specific apartment within a building complex?

Depends. The model uses Google Maps data, and so if the apartment is classified as a whole building in Google Maps, then the roof of a specific apartment of that building are displayed as separate.

Why does the API sometimes not return any data for a building which I know exists?

This can happen when there are newer buildings that did not exist when mapping was realized at the time.

How can I use the flux solar data layer for visualizing shade?

- North-facing roofs get less flux (solar irradiance), which means they will have higher shade. You can leverage this information when performing solar assessments. The flux data layer lets you visualize flux in colors.
- You can alternatively make your own flux tiles and customize their colors, especially if you wish to locate specific numerical values of shade on a roof.

How can I use **hourlyShadeURLs** from the dataLayers service?

You can get back a daily snapshot of shade throughout the year, and produce an animated GIF or video of shade changing during a year.

Can I perform my own custom calculations?

Yes, can use the flux and DSM (height map at each point) solar data layers for greater customization.

Why does the API return a URL, and what can I do with them?

The URLs are short-lived pointers to enable you to download the corresponding data layers.

Was this helpful?

Recommended for you

Solar API Concepts

The Solar API provides solar potential data through the buildingInsights and dataLayers endpoints. To use Solar API data, understanding the following concepts may be helpful: A building's solar potential is largely based on the amount of sunlight

Updated Oct 5, 2023

Solar API coverage

The Solar API provides solar data for hundreds of millions of buildings across the world. Imagery quality can be HIGH, MEDIUM, or LOW. The Google Maps Platform team is constantly working to improve coverage for our API services. The map below shows

Updated Oct 5, 2023

Compiling the Utilities

If the precompiled utilities don't suit your needs, you can build the WebP utilities yourself. Download libwebp-1.3.2.tar.gz from the downloads list and extract its contents. From the libwebp-1.3.2 directory, run: To see additional options, run: The

Updated Sep 14, 2023

Google Maps Platform FAQ

 developers.google.com/maps/faq

Getting Started

- What is the Google Maps Platform?
- How do I get started with Google Maps Platform?
- Which API do I need?
- How do I start using the APIs on my site?
- Which countries does the Google Maps Platform cover?
- Can I put Google Maps on my site without using Google Maps Platform products?
- How do I deliver Maps applications on mobile devices?
- Which web browsers do the Maps JavaScript API and Maps Embed API support?
- Can I print maps from the Maps JavaScript API?
- How can I be notified when there are changes to Google Maps Platform products?
- How do I contact technical support?
- When is technical support available?
- How do I recover access to my Google Account?
- How do I recover access to a specific project?
- Can I use the Maps and Places SDK for iOS on Arm-based Macs?

Understanding the terms of service

- What are the terms of service for Google Maps Platform products?
- Does my site meet the Google Maps Platform Terms of Service?
- Can I directly access map tiles and satellite imagery?
- Can I use Google Maps Platform products for tracking applications?
- Can I use Google Maps Platform products in my non-Web application?
- Can I use Google Maps Platform products on a site that is password protected?
- Can I create an application that includes Google Maps Platform data in a document?
- How can I opt out of including my content in Google search results?
- Can I generate a map image using the Maps Static API which I store and serve from my website?

Usage limits and billing

- How do I set up billing for my project?
- How do Google Cloud Platform Free Tier customers upgrade to a paid account?
- Does the Google Maps Platform have usage limits?
- How is usage cost calculated?
- How are map loads counted on the Google Maps Platform?
- How do I monitor my quota usage?
- What happens if I exceed the usage limits?
- My site gets a lot of traffic. Can I use Google Maps Platform products?
- If my web site or application becomes suddenly popular, will my maps stop working?

- How will usage be calculated and billed?
- How much does it cost to use the Google Maps Platform?
- Is pricing available in other currencies?
- I've set up billing. How do I view my bill?
- How do I avoid a large bill if my usage unexpectedly increases?
- I got a message saying that my project is linked to the "Google Maps Platform Transition Account", but I don't have access to that account. What do I do?
- Why is my quota limit set to 1 request per day? How can I raise this limit?
- I received a billing violation notice. How do I resolve this?

Using the Google Maps Platform

Errors and troubleshooting

Google Maps Platform Services

Maps JavaScript API

How long will the Maps JavaScript API work after it has been loaded?

Google Maps SDK for iOS

Google Maps SDK for Android

URL signing

Getting Started

What is the Google Maps Platform?

The Google Maps Platform is a set of APIs and SDKs that allows developers to embed Google Maps into mobile apps and web pages, or to retrieve data from Google Maps. There are several offerings. Depending on your needs, you may find yourself using one or a combination of these APIs and SDKs:

Maps:

- Maps JavaScript API
- Maps SDK for Android
- Maps SDK for iOS
- Aerial View API
- Maps Static API
- Street View Static API
- Maps URLs
- Maps Embed API

Routes:

- Routes API
- Roads API
- Directions API
- Distance Matrix API

Places:

- Places API
- Places SDK for Android
- Places SDK for iOS
- Places Library, Maps JavaScript API
- Address Validation API
- Geocoding API
- Geolocation API
- Time Zone API

Environment:

- Air Quality API
- Solar API

How do I get started with Google Maps Platform?

See [Get Started with Google Maps Platform](#).

Which API do I need?

For help in finding the right API based on your functional requirements, take a look at the [API picker](#).

How do I start using the APIs on my site?

See the [Overview](#), [Developer](#), and [Get Started](#) guides for the specific API or SDK you are interested in. For example, check out the guides for [Maps SDK for Android](#) or [Maps JavaScript API](#).

Which countries does the Google Maps Platform cover?

The Google Maps team is constantly pushing new map data out and increasing our international coverage. Consult the [Google Maps coverage data](#) for the latest coverage information. You can filter the data with the filter box at the top of the page. Please note that coverage data can change if licensing agreements with the data providers change.

Also see:

- [How can I get Google Maps Platform products to display in a language other than English?](#)
- [In which countries are transit directions available?](#)

Can I put Google Maps on my site without using Google Maps Platform products?

Yes. Google Maps now offers the ability to embed the map that you're viewing into your website or blog, without any programming or use of the Google Maps Platform. More information is available [here](#).

How do I deliver Maps applications on mobile devices?

To incorporate maps in an Android application, use the Maps SDK for Android.

To incorporate maps in a native iOS application, use the Maps SDK for iOS.

The Maps JavaScript API has been developed to cater to mobile devices, and is suitable for browser applications targeted at both the desktop and devices that include a web browser with a full JavaScript implementation, such as the Apple iPhone.

For applications targeted at devices not suitable for using the Maps JavaScript API, the Maps Static API delivers map images in GIF, JPG, and PNG formats, including markers and polylines. Note that use of the Maps Static API outside of browser based applications requires that the map image be linked to Google Maps.

Which web browsers do the Maps JavaScript API and Maps Embed API support?

The Maps JavaScript API and Maps Embed API support the following web browsers:

Desktop

- The current version of Microsoft Edge (Windows), **excluding** IE mode.
- The two latest major stable versions of Firefox (Windows, macOS, Linux).
- The two latest major stable versions of Chrome (Windows, macOS, Linux).
- The two latest major stable versions of Safari (macOS).

Android

- The current version of Chrome on Android 4.1+.
- Chrome WebView on Android 4.4+.

iOS

- Mobile Safari on the current and previous major versions of iOS.
- UIWebView and WKWebView on the current and previous major versions of iOS.
- The current version of Chrome for iOS.

Printing from the Maps JavaScript API is not supported. This is because printing support is inconsistent across commonly used browsers.

How can I be notified when there are changes to Google Maps Platform products?

You should subscribe to the Google Maps Platform Blog for news updates across the various Google Geo developer offerings.

How do I contact technical support?

See Google Maps Platform Support and Resources for information about available support options.

When is technical support available?

The support team is available 24x5 (weekdays from Monday 9 a.m. Tokyo time to Friday 5 p.m. Pacific time) excluding regional holidays for "service unusable" issues.

How can I recover access to my Google Account?

If you lost access to your Google Account (e.g. joe@mycompany.com or joe@gmail.com), you can try restoring the account access by retrieving or resetting your password. Visit the [How to recover your Google Account or Gmail](#) article in Google Account Help.

Note: To restore access to a G Suite account, ask your Organization admin to undelete the account.

How can I recover access to a specific project?

If you lost access to the project where you manage your Google Maps Platform implementation, you can try to recover it.

If you have access to your project-associated Google Account:

- **If you know the Project Owner and have access to your Google Account:** Ask the Project Owner to add you as a Project Owner or Project Editor.
If your project is part of an Organization: Contact your Organization admins and ask them to add you as a Project Owner.
- **If you do not know any of the current Project Owners:** If you don't know who the Project Owner is, or the Project Owner is unavailable, contact the support team to explore additional options to recover the project.

If you **do not** have access to your project-associated Google Account:

- **If you lost access to your Google Account:** try to recover your username or password for your account.
- **If you cannot recover access to your Google Account:** Create a new Google Account, then contact an existing project owner and ask them to add your new Google Account to the project.

Can I use the Maps and Places SDK for iOS on Arm-based Macs?

Developing on the new Arm-based Macs is possible, however, it requires building and running on a physical iOS device. This is a temporary limitation while we look into adding more support for developing on simulators.

Understanding the terms of service

What are the terms of service for Google Maps Platform products?

The Google Maps Platform Terms of Service are available at:

<https://cloud.google.com/maps-platform/terms>

Does my site meet the Google Maps Platform Terms of Service?

You can use the Google Maps Platform within your applications as long as your site meets the Google Maps Platform Terms of Service.

However, there are some uses of the Google Maps Platform that we just don't want to see: maps that identify the places to buy illegal drugs in a city, for instance, or any other illegal activity. We also respect people's privacy, so the Google Maps Platform shouldn't be used to identify private information about individuals.

You should use your own counsel to determine whether your application complies with the Google Maps Platform Terms of Service before you develop and launch it. Google engineers can only offer technical assistance and are not qualified to offer legal advice. Google reserves the right to suspend or terminate your use of the service at any time, so please read the Maps APIs Terms carefully.

Can I directly access map tiles and satellite imagery?

You may not access map tiles or satellite imagery through any mechanism besides the Google Maps Platform (such as the creation of your own mapping API or the use of a bulk tile download script). Your application's access to the tiles will be blocked if it accesses them outside of the Google Maps Platform. See the Google Maps Platform Terms of Service for more details.

Can I use Google Maps Platform products for tracking applications?

There is no restriction on displaying real-time data (tracking) with Google Maps Platform products provided that the application complies with the Google Maps Platform Terms of Service.

Can I use Google Maps in my non-Web application?

Yes, Google Maps Platform products can now be used in non-Web applications, provided that they adhere to the other restrictions of the Google Maps Platform Terms of Service.

The Maps JavaScript API is only supported when run in one of the supported browsers.

Can I use Google Maps Platform products on a site that is password protected?

Yes, Google Maps, Routes, Places, and Environment services can be used with private-access applications. See the Google Maps Platform Terms of Service for more details.

Can I create an application that includes Google Maps Platform data in a document?

If your application generates a document, either in electronic or printed form, no data from Google Maps Platform, including images, may be included in the document. Please see the Google Maps Platform Terms of Service "No Scraping" section for more details.

How can I opt out of including my content in Google search results?

We are no longer collecting this data. The use of the `indexing` parameter has been deprecated and has no effect. You no longer need to opt out explicitly, but we encourage you to remove this parameter at your earliest convenience.

To remove your page or site from search results, follow the instructions provided in our [webmaster help center](#).

Can I generate a map image using the Maps Static API which I store and serve from my website?

You may not store and serve copies of images generated using the Maps Static API from your website. All web pages that require static images must link the `src` attribute of an HTML `img` tag or the CSS `background-image` attribute of an HTML `div` tag directly to the Maps Static API so that all map images are displayed within the HTML content of the web page and served directly to end users by Google.

Usage limits and billing

How do I set up billing for my project?

See [Get Started with Google Maps Platform](#).

How do Google Cloud Platform Free Tier customers upgrade to a paid account?

The Google Cloud Platform Free Tier program provides customers a no-charge trial with a \$300 credit to use with any Google Cloud Platform (GCP) service, including the Google Maps Platform (GMP) APIs. When the no-charge trial ends, you must upgrade to a paid account to continue using these services. To upgrade to a paid account, visit the [Cloud Console](#).

Does the Google Maps Platform have usage limits?

There are no maximum daily limits on the number of requests you can make to Google Maps Platform products, and the only usage limits are related to the maximum number of queries per second (QPS) or queries per minute (QPM).

For Distance Matrix, the limit is set in events per second (EPS) calculated as the sum of client-side and server-side queries.

For Routes:Compute Route Matrix, the limit is in elements per minute (EPM), where the number of elements in a request is equal to: (number of origins × number of destinations).

The following table shows the usage limit for each API.

API	Usage limit
Address Validation	6,000 QPM
Aerial View: Lookup Video	180 QPM and 100,000 QPD
Aerial View: Render Video	100 QPM and 100 QPD
Air Quality	6,000 QPM
Directions	3,000 QPM

API	Usage limit
Distance Matrix	60,000 EPM
Dynamic Maps	30,000 QPM
Elevation	6,000 QPM
Geocoding	3,000 QPM
Geolocation	6,000 QPM
Places	6,000 QPM
Roads	30,000 QPM
Routes: Compute Routes	3,000 QPM
Routes: Compute Route Matrix	3,000 EPM
Solar	600 QPM
Static Maps	30,000 QPM
Street View Image API	30,000 QPM
Time Zone	30,000 QPM

In order to govern expenditures, you can monitor your API usage, and set daily limits to all requests to any billable API.

Google Maps Platform products must be deployed in compliance with the standard Google Maps Platform Terms of Service.

How is usage cost calculated?

For an overview of pricing for the Google Maps Platform products, please see the Pricing Sheet.

To learn more about how Google Maps Platform APIs are billed, please see Understanding billing for Maps, Routes, Places, and Environment.

How are map loads counted on the Google Maps Platform?

A single map load is charged when any of the following occur:

- A web page or application displays a map using the Maps JavaScript API.
- An application requests a single map image from the Maps Static API.

Street View panoramas are charged separately from map loads:

- A static Street View panorama is charged for each request to the Street View Static API to embed a static (non-interactive) Street View panorama.
- A dynamic Street View panorama is charged for each instantiation of a panorama object in a Maps JavaScript API, Maps SDK for Android, or Maps SDK for iOS application.

After a web page or application loads a map, or a static map image, or a Street View panorama, any user interactions with it, such as panning, zooming, or switching map layers, do not generate additional map loads or affect usage limits.

Adding a marker will not generate additional map loads, but may generate charges around how the pin location was determined (such as loading or reloading the `google.maps.Map()` class.)

How do I monitor my usage?

You can monitor the usage of individual APIs in the Google Cloud Console.

1. Select the project that contains the API you want to review.
2. From the list of APIs on the Dashboard, click the name of the API.
3. Near the top of the page, click **Metrics** or **Quotas**.

To see a traffic report and billing information for an entire project, follow these steps:

1. If you haven't already done so, set up billing.
2. Go to the Cloud Console billing page.
3. Select a project.
4. In the left sidebar, click **Reports**. Use the filters on the right sidebar to view reports on your billing account.

To learn more, see Google Maps Platform Reporting as well as Monitoring your API Usage and Capping API Usage.

What happens if I exceed the request rate (QPS) limits?

If you exceed the QPS limits of a given Google Maps Platform product, the API will return an error message. If you repeatedly exceed the limits, your access to the API may be temporarily blocked.

If you exceed the request QPS limits or otherwise abuse the service, requests will return a specific error message. If you continue to exceed limits, your access to the Google Maps Platform may be blocked.

Note: Four of the web service APIs have an equivalent client-side service available in the Maps JavaScript API: Directions, Distance Matrix, Elevation, and Geocoding.

Usage limits exceeded

If you exceed the usage limits you will get an **OVER_QUERY_LIMIT** status code as a response.

This means that the web service will stop providing normal responses and switch to returning only status code **OVER_QUERY_LIMIT** until more usage is allowed again. This can happen within a few seconds, if the error was received because your application sent too many requests per second.

If you regularly exceed your QPS usage limits, consider lowering usage, by optimizing applications to use Google Maps Platform products more efficiently. See the Optimization Guide for more information.

My site gets a lot of traffic. Can I use Google Maps Platform products?

Yes. However we recommend that you familiarize yourself with the usage limits for any of the Google Maps Platform that your application relies on.

If my web site or application becomes suddenly popular, will my maps stop working?

Once you have a billing account, if you exceed the no-charge, \$200-monthly usage limit and you do not have a valid payment method on your billing account (credit card, bank transfer, ...), the API ceases to function until you add a valid payment method.

How will usage be calculated and billed?

Usage is calculated at the end of each day, and priced as shown in the Pricing Sheet. At the end of every month, the total usage is charged to the payment method associated with your billing account. For more information, see Google Maps Platform Billing.

How much does it cost to use the Google Maps Platform?

See the Pricing Sheet for an overview of cost per API. If your application generates requests or map load volumes up to \$200 per month usage, your usage is not charged. Usage that exceeds the \$200 monthly credit will be charged to your billing account. For more details, see our guide to understanding Google Maps Platform billing.

Is pricing available in other currencies?

Additional currencies may be available within the console. When you select a different currency, rates will convert from the USD equivalent listed on our Pricing Sheet.

I've set up billing. How do I view my bill?

Google charges you at the beginning of each month for the previous month's activity, using your specified payment method. For details about your bill, see our guide to understanding Google Maps Platform billing.

Additional resources:

- Learn how to view your cost and payment history.
- Learn how to get an invoice or payment receipt.
- Learn how to set and manage billing alerts.

How do I avoid a large bill if my usage unexpectedly increases?

The Google Maps Platform provides ways to set daily request limits and set maximum daily billable limits. You can cap the maximum daily limit on usage to protect against unexpected increases. You can also set budget alerts to receive email notifications when the charges on the billing account reach a threshold you set.

Capping the maximum daily billable limit:

To avoid a large bill, you can set a daily cap on usage to protect against unexpected increases in use. You can change this limit in the Cloud Console by taking the following steps:

1. Go to the **APIs & Services Dashboard**.
2. Select a project if prompted.
3. Select an API from the list, then click the **Quotas** tab.
4. Click the **edit icon** next to the **"requests per day"** quota.

Alternatively, you can edit multiple quotas for multiple APIs using the IAM & admin Quotas dashboard.

Tip: Use a simple equation to determine your daily cap depending on how much you want to spend. For example: $(\text{Monthly spend} / \text{price per each}) / 30 = \text{requests per day cap}$ (for one API). For instance, on the Maps JavaScript API, 28,000 calls can be made per month within the no-charge tier. If you cap your Maps JavaScript API daily quota to 903 (ie $28,000 / 31$), you will not be charged. Note that your implementation may use multiple billable APIs, so adjust your equation as needed. Remember, a \$200 USD Google Maps Platform credit is available each month, so be sure to factor that into your calculation.

Setting and managing budget alerts:

Set up budget alerts to send email notifications to billing administrators when the charges on the billing account reach a threshold you set. Billing administrators will be sent email notifications when the estimated charges on the billing account exceed 50%, 90%, and 100% of the threshold.

Key Point: Setting a budget does *not* cap API usage. The purpose of budgets is to allow you to create alerts so that you know how your spending is trending over time. Alerts prompt you to take action to control your costs, but do not prevent the use of your services when the budget is met or exceeded. If you prefer to set a hard cap on API usage to prevent accruing costs, see Capping API Usage.

I got a message saying that my project is linked to the "Google Maps Platform Transition Account", but I don't have access to that account. What do I do?

The "Google Maps Platform Transition Account" was created to help certain customers transition to our new pay-as-you-go pricing plan. This transition account enabled Google to provide these customers with a one-time credit, so that they could continue using Google Maps Platform up to the limit of the \$200 no-charge tier. Once this limit is

exceeded, the billing account will shut down and you will lose access to the service. To avoid service interruptions, we urge you to set your own billing account, and continue to enjoy the \$200 monthly no-charge tier. To change the billing account for your project:

1. Create a new billing account (if you already have a billing account, you can skip this step).
2. Associate the billing account with your project.

Why is my quota limit set to 1 request per day? How can I raise this limit?

If you have not created and attached a billing account to your project, your Maps Platform APIs will be limited to 1 request per day. You can get higher quota by creating and attaching a billing account. To do this, see [Get Started with Google Maps Platform](#).

Once you have created and attached a billing account, your daily quota limit will be removed. You can decide to set a limit to prevent unexpected spend, in the Cloud Console.

I received a billing violation notice. How do I resolve this?

You have received this notice because Google has determined that your account has been using multiple billing accounts. This is a violation of Google's terms, as defined in section 3.2.4 of the Google Maps Platform Terms of Service. Under these terms, it is forbidden to create multiple billing accounts. To learn more, see the [Billing Account Violation FAQ](#).

Using the Google Maps Platform

Which keys or credentials should I use for different Maps products?

Each time you use Google Maps Platform products, you must include an API key to validate your request. The Google Maps Platform is available for Android, iOS or Web apps, and via HTTP web services.

API Key:

- An API key is a unique identifier that you generate using the Cloud Console.
- Example of loading an API with a key:

`&key=AIzaSyBjsINSH5x39Ks6c0_CoS1yr1Mb3cB3cVo`

Digital Signature:

A digital signature is generated using a URL signing secret provided to you by Google. Digital signatures are used with the Maps Static API and Street View Static API.

Restrictions:

- API key restrictions are optional, but we strongly recommend you restrict all API keys for greater security. See [API security best practices](#) for more details.

- You can add an application restriction to the API key. Once restricted, a key will only work on platforms that support that type of restriction. Four types of application restrictions are available. APIs enforcing the same restriction type can use the same restricted key.
 - IP addresses (individual servers) - for use with the web service APIs.
 - HTTP referrers (web sites) - for use with the web APIs.
 - Android app restriction (by package name and fingerprint) - for use with the Maps SDK for Android.
 - iOS app restriction (by iOS bundle identifier) - for use with the Maps SDK for iOS.
- You can also add an API restriction to the API key. For more information, see [Get, add, and restrict an API key](#).

The table below indicates the key/credential and application restriction for each Google Maps Platform API/SDK.

API/SDK	Credential & Application Restriction
Maps SDK for Android	API key with Android restriction ¹
Places SDK for Android	API key with Android restriction ¹
Maps SDK for iOS	API key with iOS restriction ¹
Places SDK for iOS	API key with iOS restriction ¹
Maps JavaScript API	API key with HTTP referer restriction ¹
Aerial View API	API key with IP address restriction ¹
Maps Static API	API key with HTTP referer restriction ¹ + Digital Signature ²
Street View Static API	API key with HTTP referer restriction ¹ + Digital Signature ²
Maps Embed API	API key with HTTP referer restriction ¹
Address Validation API	API key with IP address restriction ¹
Directions API	API key with IP address restriction ¹
Distance Matrix API	API key with IP address restriction ¹
Elevation API	API key with IP address restriction ¹
Geocoding API	API key with IP address restriction ¹
Geolocation API	API key with IP address restriction ¹

Places API	API key with IP address restriction ¹
Roads API	API key with IP address restriction ¹
Routes API	API key with IP address restriction ¹
Time Zone API	API key with IP address restriction ¹
Air Quality API	API key with IP address restriction ¹
Solar API	API key with IP address restriction ¹

¹ API key restrictions are optional, but we strongly recommend you restrict all API keys for greater security.

² Depending on usage, a digital signature may be required for the Maps Static API and Street View Static API. Regardless of usage, we strongly recommend that you use both an API key and a digital signature to authenticate your requests.

How do I switch my key restriction type from an HTTP referer to an IP address restriction?

Important: If you are using any of the web service APIs with an API key that has referer restrictions, your requests will fail with the error message: "API keys cannot have referer restrictions when used with this API." You should switch to using an API key with IP address restrictions.

Before you switch the API key restriction type from HTTP referer to IP address, ensure that all the APIs that use the API key support the IP restriction type. APIs of the same restriction type can use the same restricted key. If you need to enforce more than one restriction, add a separate key with the required restriction. See how to add a new API key.

Learn more about API key restrictions associated with Google Maps Platform products.

To switch an API key with HTTP referer restriction to IP address restriction, do the following:

1. Go to the Credentials page of the Cloud Console.
2. Select the project that contains the API key you want to edit.
3. On the **Credentials** page, from the list of API keys, select the name of the API key to edit the details of the key.
4. In the **Key restriction** section of the page, select "IP addresses (web servers, cron jobs, etc.)" and insert the appropriate server IP addresses, then click Save.

How do I get a new API key?

See Get Started with Google Maps Platform.

How can I find the changes introduced in each version of the Google Maps APIs?

Version change information for many of the Maps APIs are available at the following links:

You can also check which version of the Google Maps Platform a particular bug was introduced and fixed in using the Google Maps Platform Issue Tracker at:

<https://issuetracker.google.com/bookmark-groups/76561>

How can I load the API into a page asynchronously after the page has loaded?

Just specify the callback parameter when loading the API. More information and sample code can be found in the Getting Started chapter of the Maps JavaScript API documentation.

How can I get Google Maps Platform products to display in a language other than English?

By default the API will attempt to load the most appropriate language based on the users location or browser settings. Some APIs allow you to explicitly set a language when you make a request. More information on how to set the language is available in the documentation for each API:

Supported Languages:

Google often updates supported languages. This list may not be exhaustive and is subject to change.

Language Code	Language	Language Code	Language
af	Afrikaans	ja	Japanese
sq	Albanian	kn	Kannada
am	Amharic	kk	Kazakh
ar	Arabic	km	Khmer
hy	Armenian	ko	Korean
az	Azerbaijani	ky	Kyrgyz
eu	Basque	lo	Lao
be	Belarusian	lv	Latvian
bn	Bengali	lt	Lithuanian
bs	Bosnian	mk	Macedonian
bg	Bulgarian	ms	Malay
my	Burmese	ml	Malayalam

Language Code	Language	Language Code	Language
ca	Catalan	mr	Marathi
zh	Chinese	mn	Mongolian
zh-CN	Chinese (Simplified)	ne	Nepali
zh-HK	Chinese (Hong Kong)	no	Norwegian
zh-TW	Chinese (Traditional)	pl	Polish
hr	Croatian	pt	Portuguese
cs	Czech	pt-BR	Portuguese (Brazil)
da	Danish	pt-PT	Portuguese (Portugal)
nl	Dutch	pa	Punjabi
en	English	ro	Romanian
en-AU	English (Australian)	ru	Russian
en-GB	English (Great Britain)	sr	Serbian
et	Estonian	si	Sinhalese
fa	Farsi	sk	Slovak
fi	Finnish	sl	Slovenian
fil	Filipino	es	Spanish
fr	French	es-419	Spanish (Latin America)
fr-CA	French (Canada)	sw	Swahili
gl	Galician	sv	Swedish
ka	Georgian	ta	Tamil
de	German	te	Telugu
el	Greek	th	Thai

Language Code	Language	Language Code	Language
gu	Gujarati	tr	Turkish
iw	Hebrew	uk	Ukrainian
hi	Hindi	ur	Urdu
hu	Hungarian	uz	Uzbek
is	Icelandic	vi	Vietnamese
id	Indonesian	zu	Zulu
it	Italian		

You can see what the map will look like in any of the languages listed above in this [sample application](#).

Can the Google Maps Platform be accessed over SSL (HTTPS)?

The Maps JavaScript API, Maps Static API, and Web Service APIs can be accessed over secure (HTTPS) connections. Please see the documentation for the API concerned for information on how to access the API over SSL.

Note that the Maps Static API does not support custom icon URLs that use HTTPS; the default icon will be displayed.

How do I report a bug or request a new feature in the Google Maps Platform?

If you experience behavior that you believe may be a bug, please begin by raising it in the relevant forum. This will allow other developers to validate the bug, and rule out any potential issues with your code.

If you wish to request a feature, please also first raise it in the relevant forum to confirm that a solution that meets your requirements is not already available.

Once you have confirmed that you have identified a new bug, or that your requirements cannot be met by the existing functionality of the Google Maps Platform products, please report your bug or feature request using the [Google Maps Platform Issue Tracker](#).

Before adding a bug or feature request to the Issue Tracker please be sure to check that the bug or feature concerned has not already been added. If it has, you can star the issue to register your interest and be notified of updates.

How do Google Maps Platform APIs use site cookies?

Maps SDK for Android and Maps SDK for iOS use cookies subject to Google's Privacy Policy, such as calculating daily and 7-day active users and service abuse prevention. These cookies are not associated with any signed-in Google Account and are not logged

with the rest of the information collected from the API calls.

I can't find the answer to my question. Who should I contact?

Google's Developer Relations team maintains a presence on Stack Overflow — a collaboratively-edited question and answer site for programmers. It's a great place to ask technical questions about developing and maintaining Google Maps applications. More information about asking questions on Stack Overflow is available on the [Support page](#).

For best results when requesting help, please keep the following in mind:

- Search the current discussions. Chances are someone else has experienced a similar issue and found a fix.
- Submit a link to your site if possible. Only post code snippets if the code is not easily viewable online.
- Provide all relevant information including browser versions, errors, and all other facts that may be useful in troubleshooting this problem.

Errors and troubleshooting

What does this error mean?

If you encounter an error while loading or running the Google Maps APIs, please see the following links to find explanations for the error codes:

My maps appear darker than usual. What's happening?

Under certain circumstances, a darkened map, or 'negative' Street View image, watermarked with the text "for development purposes only", may be displayed. This behavior typically indicates issues with either an API key or billing. To use Google Maps Platform products, you must have a billing account and all requests must include a valid API key. The following flow will help troubleshoot this:

Are you using an API key?

Is a billing account attached to your project?

Is the provided billing method no longer valid (for example an expired credit card)?

Is there an exceeded self-imposed daily limit on the API?

Does your API key have an IP addresses restriction?

How do I resolve the error codes: **OVER_DAILY_LIMIT or **OVER_QUERY_LIMIT**?**

These error codes can be returned for any of the following reasons:

- An API key is missing from the request.
- The provided API key is invalid.
- The project does not have a billing account attached.

- A self-imposed usage cap has been exceeded.
- The provided method of payment is no longer valid (for example, a credit card has expired).
- You have exceeded the QPS limits for a given API.

To use Google Maps Platform products, you must have a billing account, and all requests must include a valid API key. To fix this, take the following steps:

How do I resolve the error codes: `kGMSPlacesRateLimitExceeded` or `9005 PLACES_API_RATE_LIMIT_EXCEEDED?`

If you are seeing `kGMSPlacesRateLimitExceeded` or `9005 PLACES_API_RATE_LIMIT_EXCEEDED`, you may be using a deprecated version of the Places SDK for Android or Places SDK for iOS. Learn more and find the new SDKs at <https://goo.gle/places-sdk-deprecation>.

How do I resolve the error: "This IP, site or mobile application is not authorized to use this API key."?

There are various scenarios which may cause this error:

- You've enabled IP address (server) restrictions on your API key, and an unauthorized IP address is attempting a request.
- You've enabled HTTP referrer (website) restrictions on your API key, and an unauthorized referrer is attempting a request.
- You've restricted usage to your Android apps by setting a package name and fingerprint, and an unauthorized Android app is attempting a request.
- You've restricted requests from iOS apps by specifying bundle identifiers and an unrecognized iOS app is attempting to send a request.
- It used to be possible to get this error if you used any of the web service APIs, with an API key with HTTP referer restrictions. Requests to those APIs should be identified with an API key with **IP address** restrictions. Switch your key restriction type from an HTTP referer restriction to an IP address restriction. For more information about restricting API keys, see [API Key Best Practices](#).

View and Edit your API Key Credentials

To view your API keys and manage any restrictions, do the following:

1. Go to the Credentials page of the Cloud Console.
2. Select the project that contains the API key you want to review.
3. To view credential details, including any restrictions set for the key, from the list of API keys, click the name of the key.
4. The full credentials of the selected API key are displayed, including any restrictions set up for the key. From here, the restrictions can be changed, deleted, or updated as needed.

How do I resolve the error: "API keys with referer restrictions cannot be used with this API."?

You are using any of the web service APIs with an API key restricted to an HTTP referer. For security reasons, web service APIs need to use API keys restricted to **IP addresses**. Switch your key restriction type from an HTTP referer restriction to an IP address restriction, or create a new API key if your key is already used with the Maps JavaScript API.

Google Maps Platform Services

I need to convert addresses to latitude/longitude pairs. Can I do that with the Google Maps Platform?

Yes, this process is called "geocoding." The Maps JavaScript API includes a class for performing a geocoding service. The class is: `google.maps.Geocoder`.

Alternatively, Google also provides the Geocoding API, which offers a REST interface that can respond in JSON and XML formats.

Which countries is geocoding available in?

To see countries currently supported by the Google Maps Platform geocoders, please consult the Google Maps coverage data.

The accuracy of geocoded locations may vary per country, so you should consider using the returned `location_type` field to determine if a good enough match has been found for the purposes of your application. Please note that the availability of geocoding data depends on our contracts with data providers, so it is subject to change.

Why do the Google Maps Platform Geocoders provide different locations than Google Maps?

The API geocoder and Google Maps geocoder sometimes use different data sets (depending on the country). The API geocoder occasionally gets updated with new data, so you can expect to see results changing or improving over time.

How should I format my geocoder queries to maximise the number of successful requests?

The geocoder is designed to map street addresses to geographical coordinates. We therefore recommend that you format geocoder requests in accordance with the following guidelines to maximize the likelihood of a successful query:

- Specify addresses in accordance with the format used by the national postal service of the country concerned.
- Do not specify additional address elements such as business names, unit numbers, floor numbers, or suite numbers that are not included in the address as defined by the postal service of the country concerned. Doing so may result in responses with **ZERO_RESULTS**.

- Format plus codes as shown here (plus signs are url-escaped to `%2B` and spaces are url-escaped to `%20`):
 - **global code** is a 4 character area code and 6 character or longer local code (849VCWC8+R9 is `849VCWC8%2BR9`).
 - **compound code** is a 6 character or longer local code with an explicit location (CWC8+R9 Mountain View, CA, USA is `CWC8%2BR9%20Mountain%20View%20CA%20USA`).
- Use the street number of a premise in preference to the building name where possible.
- Use street number addressing in preference to specifying cross streets where possible.
- Do not provide 'hints' such as nearby landmarks.

How should I format a U.S. address on a numbered highway for geocoding?

The Google Maps Platform geocoder requires that U.S. numbered highways be specified in addresses as follows:

- **County Roads:** "`Co Road NNN`" where *NNN* is the road number. eg. "`Co Road 82`"
- **State Highways:** "`State NNN`" where *State* is the full name of the state and *NNN* is the highway number. eg. "`California 82`"
- **U.S. Highways:** "`U.S. NNN`" where *NNN* is the highway number. eg. "`U.S. 101`"
- **U.S. Interstates:** "`Interstate NNN`" where *NNN* is the interstate number. eg. "`Interstate 280`"

When should I use an API geocoder class and when should I use the HTTP Geocoding Service?

See the document: Geocoding Strategies, which details the pros and cons of different geocoding strategies.

How do I provide driving directions with the Google Maps Platform?

The Compute Routes service of the Routes API and the Directions API allow you to provide driving directions for single and multi-leg journeys. Routing options help you shape directions with a travel mode (driving), a single or a set of routes, and restrictions (no toll roads). These services are available in the following forms:

- **HTTP request/response interface** (used in mobile and other applications) is compatible with Maps SDK for Android and with other Google Maps web services.
- **JavaScript API**, for client-side applications, allows you to provide driving directions via the `google.maps.DirectionsService` class. The `DirectionsRenderer` class can automatically create the overlays and directions panel for you. Additional examples are provided in the documentation.
- **Java, Python, go, and Node.js client interfaces**, for server-side applications, provide the same functionality. For more information on client libraries, see Directions API client libraries.

In which countries are driving directions available?

To see countries currently supported by driving directions in the Google Maps Platform products, consult the Google Maps coverage data. Please note that the availability of driving directions data depends on our contracts with data providers, and is subject to change.

In which countries are transit directions available?

The Directions API and Distance Matrix API support all Google Transit partners, except those in Japan.

Which KML and GeoRSS features are supported in the Maps JavaScript API?

The `KmlLayer` class in the Maps JavaScript API enables developers to overlay KML/KMZ and GeoRSS files on top of the map. Documentation and examples can be found [here](#).

What are the limits on the size and complexity of KML that can be displayed using the `KmlLayer` class of the Maps JavaScript API?

The size and complexity limits on the display of KML using the `KmlLayer` class are documented [here](#).

How do I render KML files that are hosted on intranet sites on a map?

The `KmlLayer` class that generates KML overlays in the Maps JavaScript API uses a Google hosted service to retrieve and parse KML files for rendering. Consequently it is not possible to display KML files that are not hosted at a URL that is available publicly accessible, or that require authentication to access.

If you need to develop applications that use KML files hosted on intranet sites we recommend that you render the KML on the client side by using third-party JavaScript libraries. As the KML file is analyzed by the browser, performance may be lower than by using the `KmlLayer` class.

What is the maximum number of markers or path vertices supported by the Maps Static API?

There is no limit to the number of markers or path vertices supported by the Maps Static API. When using custom icons, up to five unique icons can be specified per request, but each can be used multiple times within the map.

Note that Maps Static API URLs can contain a maximum of approximately 8,192 characters which constrains the number of markers and path vertices that can be specified based on the number of decimal places used when specifying each latitude/longitude pair. For information on how the number of decimal places used relates to the accuracy on the Earth see the Wikipedia article on [Decimal Degrees](#).

Why can't I access Google Maps Platform products for certain countries?

Maps APIs may not be used in Prohibited Territories. Refer also to the [Terms of Service](#).

How do I report a problem on the Google basemap?

Send feedback through Google Maps for wrong or missing map information such as:

- Wrong addresses or marker locations
- Incorrect road names
- Wrong information about one-way and two-way roads
- Incorrectly drawn road
- Closed roads
- Roads that don't exist

For correction of a place or business listing, suggest an edit.

If Maps content needs to be removed for legal reasons, submit a legal request.

For critical or time-sensitive requests, file a support case with specific details on what needs to be fixed.

How is performance monitored for the services used with Maps JavaScript API?

Some client-side features are instrumented to report success or failure for the purpose of calculating the SLO (Service Level Objective). This information is sent to Google at maps.googleapis.com/maps_api_js_slo/log in calls that log SLO information. This information includes success status, latency, and version/channel of the Maps JavaScript API in use. The calls may be batched for performance. Please note that you may need to allow maps.googleapis.com in your Content Security Policy to ensure these calls are not blocked at browser-level. For example: `Content-Security-Policy: default-src 'self' maps.googleapis.com;` with HTTP headers, or `<meta http-equiv="Content-Security-Policy" content="default-src 'self' maps.googleapis.com;">` with HTML Meta Tags.

Maps JavaScript API

How long will the Maps JavaScript API work after it has been loaded?

You need to refresh the page that loads the Maps JavaScript API at least once every 5 days.

Google Maps SDK for iOS

How do I resolve the error: `kGMSPlacesRateLimitExceeded`?

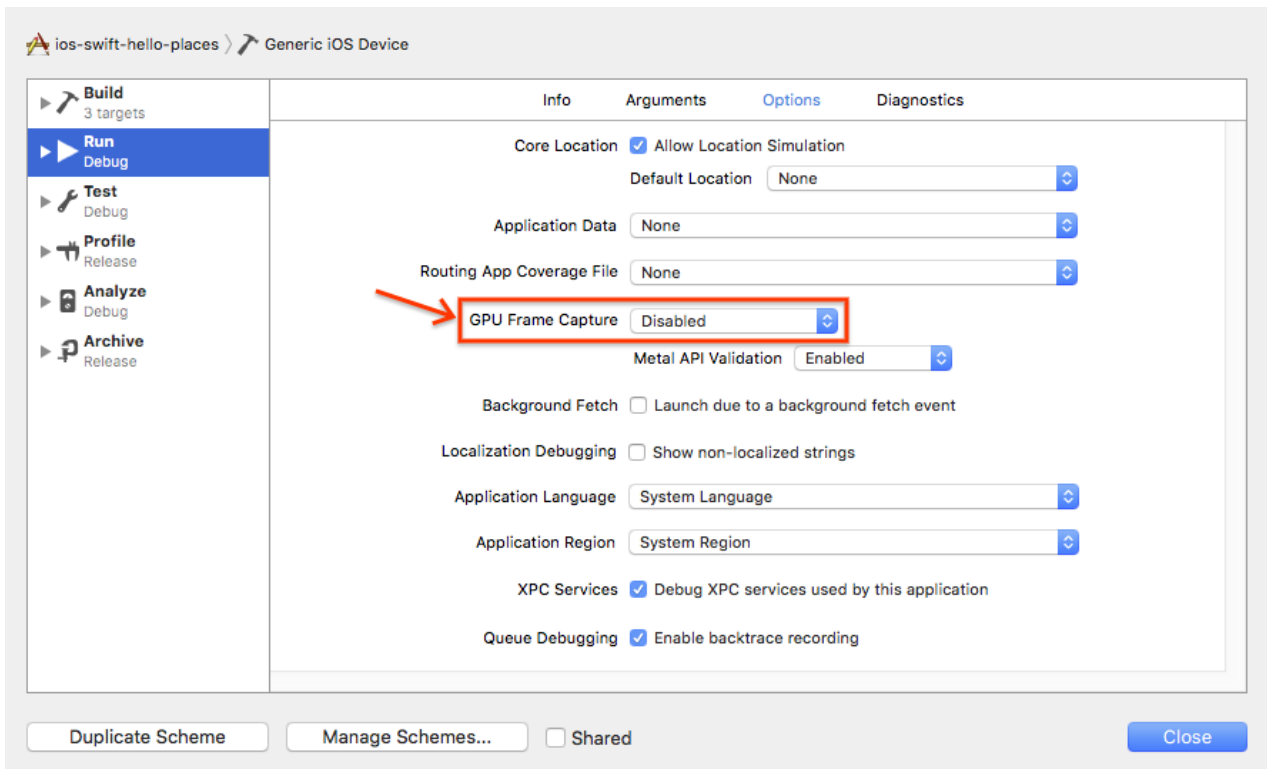
If you are seeing this error, you may be using a deprecated version of the Places SDK for iOS. Version 2.7.0 of the Places SDK for iOS has been turned off, and is no longer available. Please update your app to use the latest version as soon as possible. See the migration guide for details.

I'm getting crashes when debugging with an iOS 8.x device. What should I do?

If you experience issues when debugging with devices running iOS 8.x, follow these steps to disable GPU frame capture in the XCode scheme you are running:

1. In XCode, choose **Product, Scheme, Manage Schemes**.
2. Select a scheme and choose **Edit...**

- Set the **GPU Frame Capture** option to **Disabled**. Note that this option may not be present for all devices.



What's the impact of rounded and wider screens on the Google Maps iOS SDK?

Beginning with the iPhone X, iPhones have a screen shape featuring rounded corners, a notch at the top for the device's sensor housing, and an indicator at the bottom of the screen for accessing the Home screen. As of iOS 11.0 SDK, Apple added the Safe Area API allowing developers to position elements in an area which is safe from being clipped by the new screen shape.

The Google Maps iOS SDK has many visual elements and controls, such as an indoor picker and the report a problem link. With version 2.4, these visual elements and controls could be clipped by the new screen shape. In landscape mode, the indoor floor picker may be clipped by either the notch or the home button indicator.

With the Google Maps iOS SDK 2.5 release these layout issues are fixed automatically. When your app is used on an iPhone X or later, the padding on `GMSMapView` and `GMSPanoramaView` are increased so that the visual elements aren't clipped.

By default, we will always include your padding. The Google Maps iOS SDK assumes that padding is intended to pad from within the safe area. If you design your interface with the assumption that everything is positioned within the safe area, this will work without any extra effort.

If you have designed an interface which doesn't work with our default adjustment, we have introduced a new property to `GMSMapView`, called `paddingAdjustmentBehavior`. `GMSMapView` now allows you to pick from one of three padding adjustments behaviors: 'Always' (default), 'Automatic', and 'Never'.

If `GMSMapView` is set to use the 'Always' padding behavior, it will always add the safe area insets to the padding. This allows you to design your interface with the assumption that all placement is from the edges of the safe area. This is the default value.

If `GMSMapView` is set to use the 'Automatic' padding behavior, it will always choose the larger of padding or safe area inset. This will allow you to add padding from the screen edge while always ensuring that all elements stay within the safe area.

If `GMSMapView` is set to use the 'Never' padding behavior, it will never add the safe area insets to the padding. This is the behavior prior to the 2.5 release and may be useful if your padding already takes into consideration the safe area, or if our other behaviors don't act well with your interface. It is your responsibility to ensure that the Google logo and copyright notices are always visible, as specified in the Google Maps Platform Terms of Service.

In contrast, `GMSPanoramaView` doesn't have an optional padding property. This means that there is no padding to adjust and `GMSPanoramaView` will always apply any necessary padding to ensure that all visual elements are within the safe area.

Google Maps SDK for Android

How do I resolve the error: 9005 PLACES_API_RATE_LIMIT_EXCEEDED?

If you are seeing this error, you may be using a deprecated version of the Places SDK for Android. The Google Play services version of the Places SDK for Android has been turned off, and is no longer available. Please update your app to use the latest version as soon as possible. See the migration guide for details.

My app only shows blank grey tiles instead of a map. How can I resolve this?

A common reason that a blank grey tiles are shown instead of a map is authentication issues. You can follow the steps below to troubleshoot those issues by using `adb logcat`.

1. Make sure you have `adb` installed. If not, you can follow the guide [here](#).
2. Install the app that shows the problem on a device or an Android emulator. If you use Android emulator, make sure the emulator settings has Play Store included.
3. In Android Studio terminal, run `adb logcat -e "Google Maps Android API"`. This will only print lines where the log message matches "Google Maps Android API" (Optionally, you can output the log to a text file by appending: `> logcat.txt`)
4. Reproduce the issue on your device and check for common errors such as:
 - The wrong/unintended API Key is being referenced in Manifest.
 - Billing isn't enabled on Project.
 - The SDK isn't enabled on project APIs.
 - The incorrect SHA1 fingerprint is added to the API Key restrictions.
 - The Google Play Service dependencies is not included in the `build.gradle` file.

URL signing

Can I sign URLs using JavaScript?

We strongly recommend against signing URLs using JavaScript as this would expose your URL signing secret to end users. Therefore signatures should only be generated by server side components.

Why am I receiving a HTTP 403 Forbidden response to my Maps API web service requests?

An HTTP 403 response indicates a permission issue, likely because the signature could not be verified for this request. This could be because:

1. A signature has been specified but is incorrect for this request.
2. The request specifies a Google Maps Platform Premium Plan an API key but does not specify a signature, and the service being called requires that requests made using an API key include a valid signature.
3. A signature has been specified but the associated Google Maps Platform Premium Plan API key has not been specified.

LatLng | Solar API | Google for Developers

 developers.google.com/maps/documentation/solar/reference/rest/v1/LatLng

LatLng

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

An object that represents a latitude/longitude pair. This is expressed as a pair of doubles to represent degrees latitude and degrees longitude. Unless specified otherwise, this object must conform to the WGS84 standard. Values must be within normalized ranges.

JSON representation

```
{
  "latitude":
    number,
  "longitude":
    number
}
```

Fields

latitude	number
The latitude in degrees. It must be in the range [-90.0, +90.0].	
longitude	number
The longitude in degrees. It must be in the range [-180.0, +180.0].	

Method: geoTiff.get

 developers.google.com/maps/documentation/solar/reference/rest/v1/geoTiff/get

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

Returns an image by its ID.

HTTP request

GET <https://solar.googleapis.com/v1/geoTiff:get>

The URL uses gRPC Transcoding syntax.

Query parameters

Parameters

id **string**

The ID of the asset being requested.

Request body

The request body must be empty.

Response body

Message that represents an arbitrary HTTP body. It should only be used for payload formats that can't be represented as JSON, such as raw binary or an HTML page.

This message can be used both in streaming and non-streaming API methods in the request as well as the response.

It can be used as a top-level request field, which is convenient if one wants to extract parameters from either the URL or HTTP template into the request fields and also want access to the raw HTTP body.

Example:

```

message GetResourceRequest {
  // A unique request id.
  string request_id = 1;

  // The raw HTTP body is bound to this field.
  google.api.HttpBody http_body = 2;
}

service ResourceService {
  rpc GetResource(GetResourceRequest)
    returns (google.api.HttpBody);
  rpc UpdateResource(google.api.HttpBody)
    returns (google.protobuf.Empty);
}

```

Example with streaming methods:

```

service CaldavService {
  rpc GetCalendar(stream google.api.HttpBody)
    returns (stream google.api.HttpBody);
  rpc UpdateCalendar(stream google.api.HttpBody)
    returns (stream google.api.HttpBody);
}

```

Use of this type only changes how the request and response bodies are handled, all other features will continue to work unchanged.

If successful, the response is a generic HTTP response whose format is defined by the method.

JSON representation

```

{
  "contentType":
string,
  "data": string,
  "extensions": [
    {
      "@type":
string,
      field1: ...,
      ...
    }
  ]
}

```

Fields

contentTypestring

The HTTP Content-Type header value specifying the content type of the body.

datastring (bytes format)

The HTTP request/response body as raw binary.

A base64-encoded string.

extensions[]object

Application specific response metadata. Must be set in the first response for streaming APIs.

An object containing fields of an arbitrary type. An additional field "@type" contains a URI identifying the type. Example: { "id": 1234, "@type": "types.example.com/standard/id" }.

Was this helpful?

Recommended for you

Using the API

This document is intended for developers who want to write applications that can interact with the Books API. Google Books has a mission to digitize the world's book content and make it more discoverable on the Web. The Books API is a way to search

Updated Nov 4, 2022

Overview

The APIs in the Google Books API Family let you bring Google Books features to your site or application. The new Google Books API lets you perform programmatically most of the operations that you can do interactively on the Google Books website. The

Updated May 18, 2015

Getting Started

This document details the background knowledge that you need in order to use the Google Books API. This document is intended for developers who want to write applications that can interact with the Google Books API. Google Books has a vision to

Updated Feb 23, 2017

Method: dataLayers.get

 developers.google.com/maps/documentation/solar/reference/rest/v1/dataLayers/get

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

Gets solar information for a region surrounding a location. Returns an error with code `NOT_FOUND` if the location is outside the coverage area.

HTTP request

GET `https://solar.googleapis.com/v1/dataLayers:get`

The URL uses gRPC Transcoding syntax.

Query parameters

Parameters

<code>location</code>	<code>object</code> (LatLng) The longitude and latitude for the center of the region to get data for.
<code>radiusMeters</code>	<code>number</code> The radius, in meters, defining the region surrounding that centre point for which data should be returned. The limitations on this value are: <ul style="list-style-type: none">Any value up to 100m can always be specified.Values over 100m can be specified, as long as <code>radiusMeters</code> \leq <code>pixelSizeMeters</code> * 1000.However, for values over 175m, the <code>DataLayerView</code> in the request must not include monthly flux or hourly shade.
<code>view</code>	<code>enum</code> (DataLayerView) The desired subset of the data to return.

Parameters

requiredQuality

enum (ImageryQuality)

The minimum quality level allowed in the results. No result with lower quality than this will be returned. Not specifying this is equivalent to restricting to HIGH quality only.

pixelSizeMeters

number

The minimum scale, in meters per pixel, of the data to return. Values of 0.1 (the default, if this field is not set explicitly), 0.25, 0.5, and 1.0 are supported. Imagery components whose normal resolution is less than pixelSizeMeters will be returned at the resolution specified by pixelSizeMeters; imagery components whose normal resolution is equal to or greater than pixelSizeMeters will be returned at that normal resolution.

Request body

The request body must be empty.

Response body

If successful, the response body contains an instance of DataLayers.

DataLayerView

What subset of the solar information to return.

Enums

- DATA_LAYER_VIEW_UNSPECIFIED

Equivalent to FULL.
- DSM_LAYER

Get the DSM only.
- IMAGERY_LAYERS

Get the DSM, RGB, and mask.
- IMAGERY_AND_ANNUAL_FLUX_LAYERS

Get the DSM, RGB, mask, and annual flux.
- IMAGERY_AND_ALL_FLUX_LAYERS

Get the DSM, RGB, mask, annual flux, and monthly flux.

Enums

FULL_LAYERS

Get all data.

Was this helpful?

Recommended for you

Using the API

This document is intended for developers who want to write applications that can interact with the Books API. Google Books has a mission to digitize the world's book content and make it more discoverable on the Web. The Books API is a way to search

Updated Nov 4, 2022

Overview

The APIs in the Google Books API Family let you bring Google Books features to your site or application. The new Google Books API lets you perform programmatically most of the operations that you can do interactively on the Google Books website. The

Updated May 18, 2015

Getting Started

This document details the background knowledge that you need in order to use the Google Books API. This document is intended for developers who want to write applications that can interact with the Google Books API. Google Books has a vision to

Updated Feb 23, 2017

REST Resource: dataLayers

 developers.google.com/maps/documentation/solar/reference/rest/v1/dataLayers

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

Resource: DataLayers

Information about the solar potential of a region. The actual data are contained in a number of GeoTIFF files covering the requested region, for which this message contains URLs: Each string in the **DataLayers** message contains a URL from which the corresponding GeoTIFF can be fetched. These URLs are valid for a few hours after they've been generated. Most of the GeoTIFF files are at a resolution of 10cm/pixel, but the monthly flux file is at 50cm/pixel, and the hourly shade files are at 1m/pixel. If a **pixelSizeMeters** value was specified in the **GetDataLayersRequest**, then the minimum resolution in the GeoTIFF files will be that value.

JSON representation

```
{
  "imageryDate": {
    object (Date)
  },
  "imageryProcessedDate": {
    object (Date)
  },
  "dsmUrl": string,
  "rgbUrl": string,
  "maskUrl": string,
  "annualFluxUrl": string,
  "monthlyFluxUrl": string,
  "hourlyShadeUrls": [
    string
  ],
  "imageryQuality": enum
    (ImageryQuality)
}
```

Fields

imageryDate

object (Date)

When the source imagery (from which all the other data are derived) in this region was taken. It is necessarily somewhat approximate, as the images may have been taken over more than one day.

imageryProcessedDate

object (Date)

When processing was completed on this imagery.

dsmUrl

string

The URL for an image of the DSM (digital surface map) of the region. Values are in meters above EGM96 geoid (i.e., sea level). Invalid locations (where we don't have data) are stored as -9999.

rgbUrl

string

The URL for an image of RGB data (aerial photo) of the region.

maskUrl

string

The URL for the building mask image: one bit per pixel saying whether that pixel is considered to be part of a rooftop or not.

annualFluxUrl

string

The URL for the annual flux map (annual sunlight on roofs) of the region. Values are kWh/kW/year. This is *unmasked flux*: flux is computed for every location, not just building rooftops. Invalid locations are stored as -9999: locations outside our coverage area will be invalid, and a few locations inside the coverage area, where we were unable to calculate flux, will also be invalid.

Fields

monthlyFluxUrl

string

The URL for the monthly flux map (sunlight on roofs, broken down by month) of the region. Values are kWh/kW/year. The GeoTIFF pointed to by this URL will contain twelve bands, corresponding to January...December, in order.

Fields

`hourlyShadeUrls[]` `string`

Twelve URLs for hourly shade, corresponding to January...December, in order. Each GeoTIFF will contain 24 bands, corresponding to the 24 hours of the day. Each pixel is a 32 bit integer, corresponding to the (up to) 31 days of that month; a 1 bit means that the corresponding location is able to see the sun at that day, of that hour, of that month. Invalid locations are stored as -9999 (since this is negative, it has bit 31 set, and no valid value could have bit 31 set as that would correspond to the 32nd day of the month).

An example may be useful. If you want to know whether a point (at pixel location (x, y)) saw sun at 4pm on the 22nd of June you would:

1. fetch the sixth URL in this list (corresponding to June).
2. look up the 17th channel (corresponding to 4pm).
3. read the 32-bit value at (x, y).
4. read bit 21 of the value (corresponding to the 22nd of the month).
5. if that bit is a 1, then that spot saw the sun at 4pm 22 June.

More formally: Given `month` (1-12), `day` (1...month max; February has 28 days) and `hour` (0-23), the shade/sun for that month/day/hour at a position (`x`, `y`) is the bit

```
(hourly_shade[month - 1])(x, y)[hour] & (1 << (day - 1))
```

where (`x`, `y`) is spatial indexing, [`month - 1`] refers to fetching the `month - 1`st URL (indexing from zero), [`hour`] is indexing into the channels, and a final non-zero result means "sunny". There are no leap days, and DST doesn't exist (all days are 24 hours long; noon is always "standard time" noon).

`imageryQuality` `enum (ImageryQuality)`

The quality of the result's imagery.

Date

Represents a whole or partial calendar date, such as a birthday. The time of day and time zone are either specified elsewhere or are insignificant. The date is relative to the Gregorian Calendar. This can represent one of the following:

- A full date, with non-zero year, month, and day values.
- A month and day, with a zero year (for example, an anniversary).
- A year on its own, with a zero month and a zero day.
- A year and month, with a zero day (for example, a credit card expiration date).

Related types:

- `google.type.TimeOfDay`
- `google.type.DateTime`
- `google.protobuf.Timestamp`

JSON representation

```
{
  "year": integer,
  "month": integer,
  "day": integer
}
```

Fields

<code>year</code>	<code>integer</code>	Year of the date. Must be from 1 to 9999, or 0 to specify a date without a year.
<code>month</code>	<code>integer</code>	Month of a year. Must be from 1 to 12, or 0 to specify a year without a month and day.
<code>day</code>	<code>integer</code>	Day of a month. Must be from 1 to 31 and valid for the year and month, or 0 to specify a year by itself or a year and month where the day isn't significant.

ImageryQuality

The quality of the imagery used to compute some API result.

Enums

IMAGERY_QUALITY_UNSPECIFIED No quality is known.

HIGH The quality is high: this was based on high-resolution (e.g., 10cm) DSM data, typically from low-altitude aerial imagery.

MEDIUM The quality is medium: this was based on medium-resolution (e.g., 25cm) DSM data, typically from high-altitude aerial imagery.

LOW The quality is low: this was based on low-resolution (e.g., 50cm or worse) DSM data, typically from satellite imagery.

Methods

get Gets solar information for a region surrounding a location.

Was this helpful?

Recommended for you

Using the API

This document is intended for developers who want to write applications that can interact with the Books API. Google Books has a mission to digitize the world's book content and make it more discoverable on the Web. The Books API is a way to search

Updated Nov 4, 2022

Overview

The APIs in the Google Books API Family let you bring Google Books features to your site or application. The new Google Books API lets you perform programmatically most of the operations that you can do interactively on the Google Books website. The

Updated May 18, 2015

Getting Started

This document details the background knowledge that you need in order to use the Google Books API. This document is intended for developers who want to write applications that can interact with the Google Books API. Google Books has a vision to

Updated Feb 23, 2017

Method: buildingInsights.findClosest

 developers.google.com/maps/documentation/solar/reference/rest/v1/buildingInsights/findClosest

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

Locates the closest building to a query point. Returns an error with code `NOT_FOUND` if there are no buildings within approximately 50m of the query point.

HTTP request

GET `https://solar.googleapis.com/v1/buildingInsights:findClosest`

The URL uses gRPC Transcoding syntax.

Query parameters

Parameters

<code>location</code>	<code>object</code> (LatLng) The longitude and latitude from which the API looks for the nearest known building.
<code>requiredQuality</code>	<code>enum</code> (ImageryQuality) The minimum quality level allowed in the results. No result with lower quality than this will be returned. Not specifying this is equivalent to restricting to HIGH quality only.

Request body

The request body must be empty.

Response body

Response message for `Solar.FindClosestBuildingInsights`. Information about the location, dimensions, and solar potential of a building.

If successful, the response body contains data with the following structure:

JSON representation

```
{
  "name": string,
  "center": {
    object (LatLng)
  },
  "boundingBox": {
    object (LatLngBox)
  },
  "imageryDate": {
    object (Date)
  },
  "imageryProcessedDate": {
    object (Date)
  },
  "postalCode": string,
  "administrativeArea": string,
  "statisticalArea": string,
  "regionCode": string,
  "solarPotential": {
    object (SolarPotential)
  },
  "imageryQuality": enum
  (ImageryQuality)
}
```

Fields

name	string	The resource name for the building, of the format <code>building/<place ID></code> .
center	object (LatLng)	A point near the center of the building.
boundingBox	object (LatLngBox)	The bounding box of the building.
imageryDate	object (Date)	Date that the underlying imagery was acquired. This is approximate.

Fields

imageryProcessedDate **object** (Date)

When processing was completed on this imagery.

postalCode **string**

Postal code (e.g., US zip code) this building is contained by.

administrativeArea **string**

Administrative area 1 (e.g., in the US, the state) that contains this building. For example, in the US, the abbreviation might be "MA" or "CA."

statisticalArea **string**

Statistical area (e.g., US census tract) this building is in.

regionCode **string**

Region code for the country (or region) this building is in.

solarPotential **object** (SolarPotential)

Solar potential of the building.

imageryQuality **enum** (ImageryQuality)

The quality of the imagery used to compute the data for this building.

LatLngBox

A bounding box in lat/lng coordinates.

JSON representation

```
{
  "sw": {
    object (LatLng)
  },
  "ne": {
    object (LatLng)
  }
}
```

Fields

sw **object (LatLng)**

The southwest corner of the box.

ne **object (LatLng)**

The northeast corner of the box.

SolarPotential

Information about the solar potential of a building. A number of fields in this are defined in terms of "panels". The fields `panelCapacityWatts`, `panelHeightMeters`, and `panelWidthMeters` describe the parameters of the model of panel used in these calculations.

JSON representation

```
{
  "maxArrayPanelsCount": integer,
  "panelCapacityWatts": number,
  "panelHeightMeters": number,
  "panelWidthMeters": number,
  "panelLifetimeYears": integer,
  "maxArrayAreaMeters2": number,
  "maxSunshineHoursPerYear": number,
  "carbonOffsetFactorKgPerMwh": number,
  "wholeRoofStats": {
    object (SizeAndSunshineStats)
  },
  "buildingStats": {
    object (SizeAndSunshineStats)
  },
  "roofSegmentStats": [
    {
      object
      (RoofSegmentSizeAndSunshineStats)
    },
  ],
  "solarPanels": [
    {
      object (SolarPanel)
    },
  ],
  "solarPanelConfigs": [
    {
      object (SolarPanelConfig)
    },
  ],
  "financialAnalyses": [
    {
      object (FinancialAnalysis)
    },
  ],
}
```

Fields

maxArrayPanelsCount **integer**

Size of the maximum array - that is, the maximum number of panels that can fit on the roof.

Fields

panelCapacityWatts

number

Capacity, in watts, of the panel used in the calculations.

panelHeightMeters

number

Height, in meters in portrait orientation, of the panel used in the calculations.

panelWidthMeters

number

Width, in meters in portrait orientation, of the panel used in the calculations.

panelLifetimeYears

integer

The expected lifetime, in years, of the solar panels. This is used in the financial calculations.

maxArrayAreaMeters2

number

Size, in square meters, of the maximum array.

maxSunshineHoursPerYear

number

Maximum number of sunshine hours received per year, by any point on the roof. Sunshine hours are a measure of the total amount of insolation (energy) received per year. 1 sunshine hour = 1 kWh per kW (where kW refers to kW of capacity under Standard Testing Conditions).

carbonOffsetFactorKgPerMwh

number

Equivalent amount of CO2 produced per MWh of grid electricity. This is a measure of the carbon intensity of grid electricity displaced by solar electricity.

Fields

wholeRoofStats

object ([SizeAndSunshineStats](#))

Total size and sunlight quantiles for the part of the roof that was assigned to some roof segment. Despite the name, this may not include the entire building. See [buildingStats](#).

buildingStats

object ([SizeAndSunshineStats](#))

Size and sunlight quantiles for the entire building, including parts of the roof that were not assigned to some roof segment. Because the orientations of these parts are not well characterised, the roof area estimate is unreliable, but the ground area estimate is reliable. It may be that a more reliable whole building roof area can be obtained by scaling the roof area from [wholeRoofStats](#) by the ratio of the ground areas of [buildingStats](#) and [wholeRoofStats](#).

roofSegmentStats[]

object ([RoofSegmentSizeAndSunshineStats](#))

Size and sunlight quantiles for each roof segment.

solarPanels[]

object ([SolarPanel](#))

Each [SolarPanel](#) describes a single solar panel. They are listed in the order that the panel layout algorithm placed this. This is usually, though not always, in decreasing order of annual energy production.

solarPanelConfigs[]

object ([SolarPanelConfig](#))

Each [SolarPanelConfig](#) describes a different arrangement of solar panels on the roof. They are in order of increasing number of panels. The [SolarPanelConfig](#) with `panelsCount=N` is based on the first N panels in the [solarPanels](#) list.

Fields

financialAnalyses[]

object (FinancialAnalysis)

A FinancialAnalysis gives the savings from going solar assuming a given monthly bill and a given electricity provider. They are in order of increasing order of monthly bill amount. This field will be empty for buildings in areas for which the Solar API does not have enough information to perform financial computations.

SizeAndSunshineStats

Size and sunniness quantiles of a roof, or part of a roof.

JSON representation

```
{
  "areaMeters2": number,
  "sunshineQuantiles": [
    number
  ],
  "groundAreaMeters2":
number
}
```

Fields

areaMeters2

number

The area of the roof or roof segment, in m². This is the roof area (accounting for tilt), not the ground footprint area.

sunshineQuantiles[] **number**

Quantiles of the pointwise sunniness across the area. If there are N values here, this represents the (N-1)-iles. For example, if there are 5 values, then they would be the quartiles (min, 25%, 50%, 75%, max). Values are in annual kWh/kW like maxSunshineHoursPerYear.

Fields

groundAreaMeters2	number
	The ground footprint area covered by the roof or roof segment, in m^2.

RoofSegmentSizeAndSunshineStats

Information about the size and sunniness quantiles of a roof segment.

JSON representation

```
{
  "stats": {
    object (SizeAndSunshineStats)
  },
  "center": {
    object (LatLng)
  },
  "boundingBox": {
    object (LatLngBox)
  },
  "pitchDegrees": number,
  "azimuthDegrees": number,
  "planeHeightAtCenterMeters":
number
}
```

Fields

stats	object (SizeAndSunshineStats)
	Total size and sunlight quantiles for the roof segment.

center	object (LatLng)
	A point near the center of the roof segment.

boundingBox	object (LatLngBox)
	The bounding box of the roof segment.

Fields

`pitchDegrees`

`number`

Angle of the roof segment relative to the theoretical ground plane. 0 = parallel to the ground, 90 = perpendicular to the ground.

`azimuthDegrees`

`number`

Compass direction the roof segment is pointing in. 0 = North, 90 = East, 180 = South. For a "flat" roof segment (`pitchDegrees` very near 0), azimuth is not well defined, so for consistency, we define it arbitrarily to be 0 (North).

`planeHeightAtCenterMeters` `number`

The height of the roof segment plane, in meters above sea level, at the point designated by `center`. Together with the pitch, azimuth, and center location, this fully defines the roof segment plane.

SolarPanel

SolarPanel describes the position, orientation, and production of a single solar panel. See the `panelHeightMeters`, `panelWidthMeters`, and `panelCapacityWatts` fields in `SolarPotential` for information on the parameters of the panel.

JSON representation

```
{
  "center": {
    object (LatLng)
  },
  "orientation": enum
(SolarPanelOrientation),
  "yearlyEnergyDcKwh": number,
  "segmentIndex": integer
}
```

Fields

center	object (LatLng)	The centre of the panel.
orientation	enum (SolarPanelOrientation)	The orientation of the panel.
yearlyEnergyDcKwh	number	How much sunlight energy this layout captures over the course of a year, in DC kWh.
segmentIndex	integer	Index in roofSegmentStats of the RoofSegmentSizeAndSunshineStats which corresponds to the roof segment that this panel is placed on.

SolarPanelOrientation

The orientation of a solar panel. This must be interpreted relative to the azimuth of the roof segment that the panel is placed on.

Enums

SOLAR_PANEL_ORIENTATION_UNSPECIFIED	No panel orientation is known.
LANDSCAPE	A LANDSCAPE panel has its long edge perpendicular to the azimuth direction of the roof segment that it is placed on.
PORTRAIT	A PORTRAIT panel has its long edge parallel to the azimuth direction of the roof segment that it is placed on.

SolarPanelConfig

SolarPanelConfig describes a particular placement of solar panels on the roof.

JSON representation

```
{
  "panelsCount": integer,
  "yearlyEnergyDcKwh":
number,
  "roofSegmentSummaries": [
    {
      object
(RoofSegmentSummary)
    }
  ]
}
```

Fields

panelsCount	integer	Total number of panels. Note that this is redundant to (the sum of) the corresponding fields in roofSegmentSummaries.
yearlyEnergyDcKwh	number	How much sunlight energy this layout captures over the course of a year, in DC kWh, assuming the panels described above.
roofSegmentSummaries[]	object (RoofSegmentSummary)	Information about the production of each roof segment that is carrying at least one panel in this layout. roofSegmentSummaries[i] describes the i-th roof segment, including its size, expected production and orientation.

RoofSegmentSummary

Information about a roof segment on the building, with some number of panels placed on it.

JSON representation

```
{
  "panelsCount": integer,
  "yearlyEnergyDcKwh":
number,
  "pitchDegrees": number,
  "azimuthDegrees":
number,
  "segmentIndex": integer
}
```

Fields

panelsCount	integer	The total number of panels on this segment.
yearlyEnergyDcKwh	number	How much sunlight energy this part of the layout captures over the course of a year, in DC kWh, assuming the panels described above.
pitchDegrees	number	Angle of the roof segment relative to the theoretical ground plane. 0 = parallel to the ground, 90 = perpendicular to the ground.
azimuthDegrees	number	Compass direction the roof segment is pointing in. 0 = North, 90 = East, 180 = South. For a "flat" roof segment (pitchDegrees very near 0), azimuth is not well defined, so for consistency, we define it arbitrarily to be 0 (North).
segmentIndex	integer	Index in roofSegmentStats of the corresponding RoofSegmentSizeAndSunshineStats.

FinancialAnalysis

Analysis of the cost and benefits of the optimum solar layout for a particular electric bill size.

JSON representation

```
{
  "monthlyBill": {
    object (Money)
  },
  "defaultBill": boolean,
  "averageKwhPerMonth": number,
  "financialDetails": {
    object (FinancialDetails)
  },
  "leasingSavings": {
    object (LeasingSavings)
  },
  "cashPurchaseSavings": {
    object (CashPurchaseSavings)
  },
  "financedPurchaseSavings": {
    object
    (FinancedPurchaseSavings)
  },
  "panelConfigIndex": integer
}
```

Fields

monthlyBill	object (Money)	The monthly electric bill this analysis assumes.
defaultBill	boolean	Whether this is the bill size selected to be the default bill for the area this building is in. Exactly one FinancialAnalysis in BuildingSolarPotential should have defaultBill set.
averageKwhPerMonth	number	How much electricity the house uses in an average month, based on the bill size and the local electricity rates.

Fields

financialDetails	object (FinancialDetails)	Financial information that applies regardless of the financing method used.
leasingSavings	object (LeasingSavings)	Cost and benefit of leasing the solar panels.
cashPurchaseSavings	object (CashPurchaseSavings)	Cost and benefit of buying the solar panels with cash.
financedPurchaseSavings	object (FinancedPurchaseSavings)	Cost and benefit of buying the solar panels by financing the purchase.
panelConfigIndex	integer	Index in solarPanelConfigs of the optimum solar layout for this bill size. This can be -1 indicating that there is no layout. In this case, the remaining submessages will be omitted.

Money

Represents an amount of money with its currency type.

JSON representation

```
{
  "currencyCode":
string,
  "units": string,
  "nanos": integer
}
```

Fields

currencyCode	string	The three-letter currency code defined in ISO 4217.
units	string (int64 format)	The whole units of the amount. For example if currencyCode is "USD", then 1 unit is one US dollar.
nanos	integer	Number of nano (10^-9) units of the amount. The value must be between -999,999,999 and +999,999,999 inclusive. If units is positive, nanos must be positive or zero. If units is zero, nanos can be positive, zero, or negative. If units is negative, nanos must be negative or zero. For example \$-1.75 is represented as units=-1 and nanos=-750,000,000.

FinancialDetails

Details of a financial analysis. Some of these details are already stored at higher levels (e.g., out of pocket cost). Total money amounts are over a lifetime period defined by the panelLifetimeYears field in SolarPotential1. Note: The out of pocket cost of purchasing the panels is given in the outOfPocketCost field in CashPurchaseSavings.

JSON representation

```
{
  "initialAcKwhPerYear": number,

  "remainingLifetimeUtilityBill": {
    object (Money)
  },
  "federalIncentive": {
    object (Money)
  },
  "stateIncentive": {
    object (Money)
  },
  "utilityIncentive": {
    object (Money)
  },
  "lifetimeSrecTotal": {
    object (Money)
  },

  "costOfElectricityWithoutSolar": {
    object (Money)
  },
  "netMeteringAllowed": boolean,
  "solarPercentage": number,
  "percentageExportedToGrid":
number
}
```

Fields

<code>initialAcKwhPerYear</code>	<code>number</code>	How many AC kWh we think the solar panels will generate in their first year.
<code>remainingLifetimeUtilityBill</code>	<code>object (Money)</code>	Utility bill for electricity not produced by solar, for the lifetime of the panels.

Fields

federalIncentive

object (Money)

Amount of money available from federal incentives; this applies if the user buys (with or without a loan) the panels.

stateIncentive

object (Money)

Amount of money available from state incentives; this applies if the user buys (with or without a loan) the panels.

utilityIncentive

object (Money)

Amount of money available from utility incentives; this applies if the user buys (with or without a loan) the panels.

lifetimeSrecTotal

object (Money)

Amount of money the user will receive from Solar Renewable Energy Credits over the panel lifetime; this applies if the user buys (with or without a loan) the panels.

costOfElectricityWithoutSolar

object (Money)

Total cost of electricity the user would have paid over the lifetime period if they didn't install solar.

netMeteringAllowed

boolean

Whether net metering is allowed.

solarPercentage

number

Percentage (0-100) of the user's power supplied by solar. Valid for the first year but approximately correct for future years.

Fields

percentageExportedToGrid

number

The percentage (0-100) of solar electricity production we assumed was exported to the grid, based on the first quarter of production. This affects the calculations if net metering is not allowed.

LeasingSavings

Cost and benefit of leasing a particular configuration of solar panels with a particular electricity usage.

JSON representation

```
{
  "leasesAllowed":
boolean,
  "leasesSupported":
boolean,
  "annualLeasingCost": {
    object (Money)
  },
  "savings": {
    object
(SavingsOverTime)
  }
}
```

Fields

leasesAllowed

boolean

Whether leases are allowed in this jurisdiction (leases are not allowed in some states). If this field is false, then the values in this message should probably be ignored.

Fields

leasesSupported **boolean**

Whether leases are supported in this jurisdiction by the financial calculation engine. If this field is false, then the values in this message should probably be ignored. This is independent of **leasesAllowed**: in some areas leases are allowed, but under conditions that aren't handled by the financial models.

annualLeasingCost **object (Money)**

Estimated annual leasing cost.

savings **object (SavingsOverTime)**

How much is saved (or not) over the lifetime period.

SavingsOverTime

Financial information that's shared between different financing methods.

JSON representation

```
{
  "savingsYear1": {
    object (Money)
  },
  "savingsYear20": {
    object (Money)
  },
  "presentValueOfSavingsYear20":
  {
    object (Money)
  },
  "savingsLifetime": {
    object (Money)
  },
  "presentValueOfSavingsLifetime":
  {
    object (Money)
  },
  "financiallyViable": boolean
}
```

Fields

savingsYear1	object (Money)	Savings in the first year after panel installation.
savingsYear20	object (Money)	Savings in the first twenty years after panel installation.
presentValueOfSavingsYear20	object (Money)	Using the assumed discount rate, what is the present value of the cumulative 20-year savings?
savingsLifetime	object (Money)	Savings in the entire panel lifetime.
presentValueOfSavingsLifetime	object (Money)	Using the assumed discount rate, what is the present value of the cumulative lifetime savings?
financiallyViable	boolean	Indicates whether this scenario is financially viable. Will be false for scenarios with poor financial viability (e.g., money-losing).

CashPurchaseSavings

Cost and benefit of an outright purchase of a particular configuration of solar panels with a particular electricity usage.

JSON representation

```
{
  "outOfPocketCost": {
    object (Money)
  },
  "upfrontCost": {
    object (Money)
  },
  "rebateValue": {
    object (Money)
  },
  "savings": {
    object
(SavingsOverTime)
  },
  "paybackYears": number
}
```

Fields

outOfPocketCost	object (Money)	Initial cost before tax incentives: the amount that must be paid out-of-pocket. Contrast with upfrontCost, which is after tax incentives.
upfrontCost	object (Money)	Initial cost after tax incentives: it's the amount that must be paid during first year. Contrast with outOfPocketCost, which is before tax incentives.
rebateValue	object (Money)	The value of all tax rebates.
savings	object (SavingsOverTime)	How much is saved (or not) over the lifetime period.

Fields

paybackYears	number
	Number of years until payback occurs. A negative value means payback never occurs within the lifetime period.

FinancedPurchaseSavings

Cost and benefit of using a loan to buy a particular configuration of solar panels with a particular electricity usage. Initial out of pocket cost is zero in our model: the loan covers everything.

JSON representation

```
{
  "annualLoanPayment": {
    object (Money)
  },
  "rebateValue": {
    object (Money)
  },
  "loanInterestRate":
number,
  "savings": {
    object
(SavingsOverTime)
  }
}
```

Fields

annualLoanPayment	object (Money)
	Annual loan payments.

rebateValue	object (Money)
	The value of all tax rebates (including Federal Investment Tax Credit (ITC)).

Fields

loanInterestRate **number**

The interest rate on loans assumed in this set of calculations.

savings **object** (SavingsOverTime)

How much is saved (or not) over the lifetime period.

Was this helpful?

Recommended for you

Using the API

This document is intended for developers who want to write applications that can interact with the Books API. Google Books has a mission to digitize the world's book content and make it more discoverable on the Web. The Books API is a way to search

Updated Nov 4, 2022

Compiling the Utilities

If the precompiled utilities don't suit your needs, you can build the WebP utilities yourself. Download libwebp-1.3.2.tar.gz from the downloads list and extract its contents. From the libwebp-1.3.2 directory, run: To see additional options, run: The

Updated Sep 14, 2023

Overview

The APIs in the Google Books API Family let you bring Google Books features to your site or application. The new Google Books API lets you perform programmatically most of the operations that you can do interactively on the Google Books website. The

Updated May 18, 2015

REST Resource: buildingInsights

 developers.google.com/maps/documentation/solar/reference/rest/v1/buildingInsights

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

Resource

There is no persistent data associated with this resource.

Methods

findClosest Locates the closest building to a query point.

Reporting & monitoring overview

 developers.google.com/maps/documentation/solar/report-monitor

- Home
- Products
- Google Maps Platform
- Documentation
- Solar API

Was this helpful?

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

It's important to review your Google Maps Platform API usage, quota, and billing information on a regular basis. This information helps you measure API usage, stay within predefined consumption limits, and control costs through planned budgets. Reviewing this information can also alert you to any unexpected interactions that might occur between your applications and the Google Maps Platform services.

The Maps Platform provides two tools that can help you review usage, quota, and billing information:

- **Reporting:** A set of predefined visual reports that let you easily see basic API usage, quota, and billing information in the Google Cloud Console. You can quickly determine the number of API calls, see how close you are to hitting API usage quotas, and monitor billing usage over time.
- **Monitoring:** A set of tools, both in the Cloud Console and through an API, that let you monitor API usage, quota, and billing information and define alerts when any of these metrics approaches a predefined limit.

Monitoring lets you create your own customized monitoring dashboards displaying your metrics as different chart types. You can also issue alert notifications, such as emails or SMS text messages, when a metric crosses a predefined threshold.

Reporting

Reporting in the Maps Platform provides a set of predefined visual reports that let you easily see basic API usage, quota, and billing information in the Cloud Console. View reports for your Maps Platform API usage, quota, and billing numbers by using the Cloud Console.

APIs & Services reports

The Cloud Console APIs & Services report provides usage metrics for all APIs enabled for your project, including the Maps Platform APIs and SDKs as well as all other Google APIs and services.

This image shows the **APIs & Services** report.

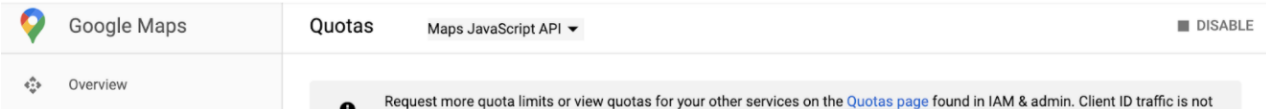
Quotas

Quotas set limits on the number of requests your project can make to the Maps Platform APIs. Requests can be limited in three ways:

- Per day
- Per minute
- Per user per minute (where available)

Only successful requests and requests that cause server errors count against quota. Requests that fail authentication do not count against quota.

Quota usage is displayed in graphs on the **Quotas** report page in the Cloud Console.



Billing

The Cloud Console Billing reports provide billing and related cost information for the project you have selected.

The Cloud Billing Reports page lets you view your Google Cloud Platform usage costs for all projects linked to a Cloud Billing account. To help you view the cost trends that are important to you, you can select a data range, specify a time range, configure the chart filters, and group by project, service, SKU, or location.

Cloud Billing reports can help you answer questions like these:

- How is my current month's Google Cloud Platform spending trending?
- What project cost the most last month?
- What are my forecasted future costs based on historical trends?
- How much am I spending by region?

This image shows the **Billing** report.

Monitoring

Cloud Monitoring collects measurements of your service and of the Google Cloud Platform resources that you use. For example, you can monitor API calls or quota usage over a specified time interval.

Not only can you define custom metrics and charts, but you can also define alerts. Use alerts to send a notification when the performance of a service doesn't meet criteria you define. For example, you can send a notification as an email, text message, to the Cloud Console Mobile App, and other options.

Metrics

In Cloud Monitoring:

- A *metric* describes something that is measured. Examples of metrics include the number of calls to an API, percent of a usage quota consumed, or a virtual machine's CPU utilization.
- A *time series* is a data structure that contains time-stamped measurements of a metric and information about the source and meaning of those measurements.

To explore metric data, build a chart with Metrics Explorer. For example, to view the request count of an API in one minute intervals for the past hour, use Metrics Explorer to construct a chart that displays the most recent data.



Dashboards

Dashboards let you view and monitor your time-series data as a collection of charts. To create custom dashboards, you can use the Cloud Console or the Cloud Monitoring API.

The following image shows a custom dashboard with two charts: a quota chart on the left, and an API count chart on the right.



Alerts

To be notified when the performance of a service doesn't meet criteria you define, create an alerting policy. For example, you can create an alerting policy that notifies your on-call team when the 90th percentile of the latency of HTTP 200 responses from your service exceeds 100 ms.

Alerting gives timely awareness to problems in your cloud applications so you can resolve the problems quickly.

Cloud Monitoring supports many types of alerts such as:

- Metric threshold alerts: Trigger an alert if a metric rises above or falls below a value for a specific duration or a metric increases or decreases by a predefined percentage.
- Budget alerts: Trigger notifications when your costs exceed a percentage of your budget.
- Quota alerts: Trigger notifications when your usage approaches a quota limit.

What's next

- Reporting
- Monitoring

Was this helpful?

Recommended for you

Using the API

This document is intended for developers who want to write applications that can interact with the Books API. Google Books has a mission to digitize the world's book content and make it more discoverable on the Web. The Books API is a way to search

Updated Nov 4, 2022

Compiling the Utilities

If the precompiled utilities don't suit your needs, you can build the WebP utilities yourself. Download libwebp-1.3.2.tar.gz from the downloads list and extract its contents. From the libwebp-1.3.2 directory, run: To see additional options, run: The

Updated Sep 14, 2023

Getting Started

This document details the background knowledge that you need in order to use the Google Books API. This document is intended for developers who want to write applications that can interact with the Google Books API. Google Books has a vision to

Updated Feb 23, 2017

Best Practices Using Solar API Web Services

 developers.google.com/maps/documentation/solar/web-service-best-practices

The Google Maps Platform web services are a collection of HTTP interfaces to Google services providing geographic data for your maps applications.

This guide describes some common practices useful for setting up your web service requests and processing service responses. Refer to the developer's guide for full documentation of the Solar API.

What is a web service?

Google Maps Platform web services are an interface for requesting Maps API data from external services and using the data within your Maps applications. These services are designed to be used in conjunction with a map, as per the License Restrictions in the Google Maps Platform Terms of Service.

The Maps APIs web services use HTTP(S) requests to specific URLs, passing URL parameters and/or JSON-format POST data as arguments to the services. Generally, these services return data in the response body as JSON for parsing and/or processing by your application.

The following example shows the URL of a REST **GET** request to the **dataLayers** method:

```
https://solar.googleapis.com/v1/dataLayers:get?parameters
```

Note: All Solar API applications require authentication. Get more information on authentication credentials.

SSL/TLS Access

HTTPS is required for all Google Maps Platform requests that use API keys or contain user data. Requests made over HTTP that contain sensitive data may be rejected.

Building a valid URL

You may think that a "valid" URL is self-evident, but that's not quite the case. A URL entered within an address bar in a browser, for example, may contain special characters (e.g. "上海+中國"); the browser needs to internally translate those characters into a different encoding before transmission. By the same token, any code that generates or accepts UTF-8 input might treat URLs with UTF-8 characters as "valid", but would also need to translate those characters before sending them out to a web server. This process is called URL-encoding or percent-encoding.

Caution: Browsers and/or services may automatically URL-encode a request URI before sending. On APIs that use cryptographic request signing, this can potentially invalidate

the signature, if URL-encoding alters the request after signing. To avoid this issue, *always* URL-encode your query string **before** signing the request.

Special characters

We need to translate special characters because all URLs need to conform to the syntax specified by the Uniform Resource Identifier (URI) specification. In effect, this means that URLs must contain only a special subset of ASCII characters: the familiar alphanumeric symbols, and some reserved characters for use as control characters within URLs. This table summarizes these characters:

Set	characters	URL usage
Alphanumeric	a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9	Text strings, scheme usage (http), port (8080), etc.
Unreserved	- _ . ~	Text strings
Reserved	! * ' () ; : @ & = + \$, / ? % # []	Control characters and/or Text Strings

When building a valid URL, you must ensure that it contains only those characters shown in the Summary of Valid URL Characters table. Conforming a URL to use this set of characters generally leads to two issues, one of omission and one of substitution:

- Characters that you wish to handle exist outside of the above set. For example, characters in foreign languages such as [上海+中國](#) need to be encoded using the above characters. By popular convention, spaces (which are not allowed within URLs) are often represented using the plus '+' character as well.
- Characters exist within the above set as reserved characters, but need to be used literally. For example, [?](#) is used within URLs to indicate the beginning of the query string; if you wish to use the string ["? and the Mysterions,"](#) you'd need to encode the ['?'](#) character.

All characters to be URL-encoded are encoded using a '%' character and a two-character hex value corresponding to their UTF-8 character. For example, [上海+中國](#) in UTF-8 would be URL-encoded as [%E4%B8%8A%E6%B5%B7%2B%E4%B8%AD%E5%9C%8B](#). The string [? and the Mysterions](#) would be URL-encoded as [%3F+and+the+Mysterions](#) or [%3F%20and%20the%20Mysterions](#).

Common characters that need encoding

Some common characters that must be encoded are:

Unsafe character Encoded value

Space	%20
"	%22
<	%3C
>	%3E
#	%23
%	%25
	%7C

Converting a URL that you receive from user input is sometimes tricky. For example, a user may enter an address as "5th&Main St." Generally, you should construct your URL from its parts, treating any user input as literal characters.

Additionally, URLs are limited to 16384 characters for all Google Maps Platform web services and static web APIs. For most services, this character limit will seldom be approached. However, note that certain services have several parameters that may result in long URLs.

Polite Use of Google APIs

Poorly designed API clients can place more load than necessary on both the Internet and Google's servers. This section contains some best practices for clients of the APIs. Following these best practices can help you avoid your application being blocked for inadvertent abuse of the APIs.

Exponential Backoff

In rare cases something may go wrong serving your request; you may receive a 4XX or 5XX HTTP response code, or the TCP connection may simply fail somewhere between your client and Google's server. Often it is worthwhile re-trying the request as the followup request may succeed when the original failed. However, it is important not to simply loop repeatedly making requests to Google's servers. This looping behavior can overload the network between your client and Google causing problems for many parties.

A better approach is to retry with increasing delays between attempts. Usually the delay is increased by a multiplicative factor with each attempt, an approach known as Exponential Backoff.

For example, consider an application that wishes to make this request to the Time Zone API:

```
https://maps.googleapis.com/maps/api/timezone/json?
location=39.6034810, -119.6822510&timestamp=1331161200&key=YOUR_API_KEY
```

The following Python example shows how to make the request with exponential backoff:

```
import json
import time
import urllib.error
import urllib.parse
import urllib.

request

# The maps_key defined below isn't a valid Google Maps API key.
# You need to get your own API key.
# See https://developers.google.com/maps/documentation/timezone/get-api-key
API_KEY = "YOUR_KEY_HERE"
TIMEZONE_BASE_URL = "https://maps.googleapis.com/maps/api/timezone/json"
def timezone(lat, lng, timestamp):
    # Join the parts of the URL together into one string.
    params = urllib.parse.urlencode(
        {"location": f"{lat},{lng}", "timestamp": timestamp, "key": API_KEY,}
    )
    url = f"{TIMEZONE_BASE_URL}?{params}"    current_delay = 0.1 # Set the
initial retry delay to 100ms.
    max_delay = 5 # Set the maximum retry delay to 5 seconds.
    while True:
        try:
            # Get the API response.
            response = urllib.request.urlopen(url)
        except urllib.error.URLError:
            pass # Fall through to the retry loop.
        else:
            # If we didn't get an IOError then parse the result.
            result = json.load(response)
            if result["status"] == "OK":
                return result["timeZoneId"]
            elif result["status"] != "UNKNOWN_ERROR":
                # Many API errors cannot be fixed by a retry, e.g.
INVALID_REQUEST or
                # ZERO_RESULTS. There is no point retrying these requests.
                raise Exception(result["error_message"])
            if current_delay > max_delay:
                raise Exception("Too many retry attempts.")
            print("Waiting", current_delay, "seconds before retrying.")
            time.sleep(current_delay)
            current_delay *= 2 # Increase the delay each time we retry.
if __name__ == "__main__":
    tz = timezone(39.6034810, -119.6822510, 1331161200)
    print(f"Timezone: {tz}")
```

You should also be careful that there isn't retry code higher in the application call chain that leads to repeated requests in quick succession.

Synchronized Requests

Large numbers of synchronized requests to Google's APIs can look like a Distributed Denial of Service (DDoS) attack on Google's infrastructure, and be treated accordingly. To avoid this, you should make sure that API requests are not synchronized between clients.

For example, consider an application that displays the time in the current time zone. This application will probably set an alarm in the client operating system waking it up at the start of the minute so that the displayed time can be updated. The application should *not* make any API calls as part of the processing associated with that alarm.

Making API calls in response to a fixed alarm is bad as it results in the API calls being synchronized to the start of the minute, even between different devices, rather than being distributed evenly over time. A poorly designed application doing this will produce a spike of traffic at sixty times normal levels at the start of each minute.

Instead, one possible good design is to have a second alarm set to a randomly chosen time. When this second alarm fires the application calls any APIs it needs and stores the results. When the application wants to update its display at the start of the minute, it uses previously stored results rather than calling the API again. With this approach, API calls are spread evenly over time. Further, the API calls do not delay rendering when the display is being updated.

Aside from the start of the minute, other common synchronization times you should be careful *not* to target are at the start of an hour, and the start of each day at midnight.

Processing Responses


Note: As the exact format of individual responses with a web service request is not guaranteed (some elements may be missing or in multiple locations), you should never assume that the format returned for any given response will be the same for different queries. Instead, you should **process** the response and select appropriate values via **expressions**.

This section discusses how to extract these values dynamically from web service responses.

The Google Maps web services provide responses which are easy to understand, but not exactly user friendly. When performing a query, rather than display a set of data, you probably want to extract a few specific values. Generally, you will want to parse responses from the web service and extract only those values which interest you.

The parsing scheme you use depends on whether you are returning output in JSON. JSON responses, being already in the form of Javascript objects, may be processed within Javascript itself on the client.

Migrate to the Solar API on Google Maps Platform

 developers.google.com/maps/documentation/solar/migration

- Home
- Products
- Google Maps Platform
- Documentation
- Solar API


Was this helpful?


bookmark_border Stay organized with collections Save and categorize content based on your preferences.

To move from Google Earth Engine Solar API to the Google Maps Platform Solar API:

1. Enable the Google Maps Platform Solar API in your cloud project.
2. Create a new key and restrict it to GMP Solar API.
3. Update your code using the step-by-step instructions below.

Side-by-side comparison

	Solar API (new) 	Earth Engine Solar API (deprecated)
Launch status	Launched	Pilot (deprecated)
Access		
Mechanism	Google Cloud account via the Cloud console, by enabling the Solar API and managing the API via the Google Maps Platform section	Google Cloud account via the Cloud console, by enabling the Earth Engine Solar API
Who	Public	Access-controlled
Level	Self-provisioned	Cloud project manual access
Authentication	API Key and OAuth	API Key
Pricing		
Strategy	Pay-as-you-go	100% discount
Tiering	Per 1000 queries, with decreasing pricing based on volume	—

	Solar API (new) 	Earth Engine Solar API (deprecated)
Endpoints	Different, per-endpoint prices	—
Cloud		
Monitoring	Cloud Monitoring under "Google Maps Platform"	Cloud Monitoring under "APIs and Services"
Quota	QPM (query per minute) and QPH (query per hour)	Annual
Logging	Cloud Logging (optional)	Cloud Logging (optional)
Billing	Cloud Billing account	-
Support	Full Google Maps Platform support with SLO/SLA	Limited, by email
API		
Hostname	https://solar.googleapis.com/v1/ (REST)	https://earthengine-solar.googleapis.com/v1/ (REST)
Methods	<ul style="list-style-type: none"> <code>buildingInsights:findClosest</code> <code>dataLayers:get</code> 	<ul style="list-style-type: none"> <code>buildings:findClosest</code> <code>solar.get</code>
Response	No changes compared to pilot	—
<code>solarInfo</code>	≤100m radius	≤100m radius
Coverage		
Area	Global	Global
Data quality	HIGH/MEDIUM	HIGH/MEDIUM
Building type	Any building that is mapped to an address AND within Solar API imagery coverage	Any building that is mapped to an address AND within Solar API imagery coverage
Terms of Service		
TOS	Google Maps Platform terms	Google Earth Engine terms

Step by step

Set up your Google Cloud project

Instructions here: Set up your Google Cloud project.

Caution: Stop at the "Enable API" part of the "Set up your Google Cloud project". Only certain roles can create a Cloud project; if you can't create a project, reach out to your organization's administrator.

You can also use an existing Cloud project. To learn more, see [Getting started with Google Maps Platform](#).

Set up your Billing Account

Instructions here: [How to manage Your billing Account](#).

You can use an existing Cloud project with an existing billing account.

Get an API Key or use OAuth token

After setting up your Google Cloud project, you must create and secure your API Key to use the Solar API as described in [Use API Keys](#). Or, you can create an OAuth token as described in [Use OAuth](#).

Use the Solar API

- Make GET requests to the new endpoints : <https://solar.googleapis.com>
- Note that some API method names have changed:
 - [buildings:findClosest](#) → [buildingInsights:findClosest](#)
 - [solarinfo:get](#) → [dataLayers:get](#)

Quick trial: Use the saved API key from prior step and replace [YOUR_API_KEY](#) in the example query below, before loading the URL in your browser:

```
https://solar.googleapis.com/v1/dataLayers:get?  
location.latitude=37.2746464&location.longitude=-121.7530949&radius\_meters=10&  
key=YOUR\_API\_KEY
```

Response for the original Preview release

For the original Preview release on May 9, 2023, the URLs in the response are in the form:

```
https://earthengine.googleapis.com/v1alpha/projects/sunroof-  
api/thumbnails/THUMBNAIL\_ID:getPixels
```

The following snippet is an example response:


```
{
  "imageryDate": {
    "year": 2015,
    "month": 8,
    "day": 8
  },
  "imageryProcessedDate": {
    "year": 2021,
    "month": 2,
    "day": 15
  },
  "dsmUrl": "https://earthengine.googleapis.com/v1alpha/projects/geo-solar-api/thumbnails/fbde33e9cd16d5fd10d19a19dc580bc1-8614f599c5c264553f821cd034d5cf32:getPixels",
  "rgbUrl": "https://earthengine.googleapis.com/v1alpha/projects/geo-solar-api/thumbnails/91ed3551f2d0abee20af35e07bd0c927-c96c59e80cf1fc1dc86cf59fc8ec86ba:getPixels",
  "maskUrl": "https://earthengine.googleapis.com/v1alpha/projects/geo-solar-api/thumbnails/e4051553dba6870c03d855ae82c30b7e-7cc8ae6ce7c73f219e3c1924e5c17fc6:getPixels",
  "annualFluxUrl": "https://earthengine.googleapis.com/v1alpha/projects/geo-solar-api/thumbnails/9b0f87f49d778a65c9e27ff936e6dbba-b90be2fe80d25abd4c9e8c4dc809f763:getPixels",
  "monthlyFluxUrl": "https://earthengine.googleapis.com/v1alpha/projects/geo-solar-api/thumbnails/90e7cca77402f14809e349937f0a0be8-94fafeb4ef42d72f1b3c0652a1cb5518:getPixels",
  "hourlyShadeUrls": [
    "https://earthengine.googleapis.com/v1alpha/projects/geo-solar-api/thumbnails/dcd276e4782aef4ff1b230b781736d37-e193b231ce57a03449afc3e21cf6783b:getPixels",
    ...
  ]
}
```

To make a request to a URL in the response, include the entire URL in the request.

The full specification of this request and response is in the reference documentation.

Write an app to support both response formats

You can now write an app that handles both the original Preview and current response formats.

The main difference between the two responses, other than the actual URL itself, is that you **must** pass an API key to a request that accesses the URLs from the **new** response format. If you omit the API key, the request fails.

Note: Do not pass an API key to a URL in the original format, meaning one using the <https://earthengine.googleapis.com> domain. Passing an API key to that URL causes an error.

For example, you can add the following code to your app to examine the URL and handle each version correctly:

JavaScriptPythonJava

```
/**
 * Function to examine a response URL and to append the API key to the
 * URL if it is in the new format.
 */
function prepareGetGeoTiffUrl(geoTiffUrl, apiKey) {
  if (geoTiffUrl.match("solar.googleapis.com")) {
    let url = new URL(geoTiffUrl);
    url.searchParams.set('apiKey', apiKey);
    return url.toString();
  }
  return geoTiffUrl;
}
```

Monitor

Project level

Billing Account level

Good-to-know tips

- Quota: consumption that can scale (rather than annual which will disappear)
 - Current quota which will be changed to QPM
 - Best practices: set client-side quota and send alerts
- Pricing:
 - Pay-as-you-go
 - 404 NOT_FOUND responses, when location is not in coverage range, will *not* be billed but will count against quota
- General usage Terms: Google Maps Platform Terms of Service

Was this helpful?

Recommended for you

Solar API Usage and Billing

The Solar API uses a pay-as-you-go pricing model. Solar API requests generate calls to one SKU for all but mobile-native apps. Along with the overall Google Terms of Use, there are usage limits specific to the Solar API. Manage your costs and usage

Updated Oct 5, 2023

Using the API

This document is intended for developers who want to write applications that can interact with the Books API. Google Books has a mission to digitize the world's book content and make it more discoverable on the Web. The Books API is a way to search

Updated Nov 4, 2022

Compiling the Utilities

If the precompiled utilities don't suit your needs, you can build the WebP utilities yourself. Download libwebp-1.3.2.tar.gz from the downloads list and extract its contents. From the libwebp-1.3.2 directory, run: To see additional options, run: The

Updated Sep 14, 2023

About GeoTIFF files

developers.google.com/maps/documentation/solar/geotiff

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

The dataLayers endpoint returns data encoded as GeoTIFF files, which can be used in any geographic information system (GIS) application to design solar systems.

Each string in the dataLayers response contains a URL, which you can use to fetch the corresponding GeoTIFF. URLs are valid for up to an hour after they are generated from the original data layers request. GeoTIFF files can be stored for up to 30 days.

With the exception of the RGB layer, GeoTIFF files don't display correctly with an image viewer, as the content is encoded data rather than RGB images. GeoTIFF files also cannot be used directly as an overlay image with Maps Javascript API.

The following table describes each layer in detail.

Layer	Pixel depth	Resolution	Description
Digital Surface Model (DSM)	32-bit float	0.1 m/pixel	Elevation data that represents the topography of Earth's surface, including natural and built features. Values are in meters above sea level. Invalid locations, or areas where we don't have data, are stored as -9999.
RGB	8-bit	0.1 m/pixel 0.25 m/pixel 0.5 m/pixel 1 m/pixel	An aerial image of the region. The GeoTIFF imagery file contains three bands corresponding to red, green and blue values in order to form 24-bit RGB value for each pixel. By default, the pixel resolution is 0.1 m/pixel.
Building mask	1-bit	0.1 m/pixel	One bit per pixel indicating whether that pixel is considered to be part of a rooftop.

Layer	Pixel depth	Resolution	Description
Annual flux	32-bit float	0.1 m/pixel	<p>The annual flux map, or annual sunlight on roofs, of the region. Values are kWh/kW/year.</p> <p>Flux is computed for every location, not just building rooftops. Invalid locations, or areas where we couldn't calculate flux, are stored as -9999. Locations outside our coverage area are invalid.</p> <p>Note: This is unmasked flux.</p>
Monthly flux	32-bit float	0.5 m/pixel	<p>The monthly flux map (sunlight on roofs, broken down by month) of the region. Values are kWh/kW/year. The GeoTIFF imagery file contains 12 bands corresponding to January — December, in order.</p>
Hourly shade	32-bit integer	1 m/pixel	<p>12 URLs for hourly shade maps corresponding to January — December, in order.</p> <p>Each GeoTIFF file contains 24 bands, corresponding to the 24 hours of the day. Each pixel is a 32 bit integer, corresponding to the (up to) 31 days of that month. A 1 bit means that the corresponding location is able to see the sun on that day, at that hour, in that month.</p> <p>Invalid locations are stored as -9999 and have bit 31 set, as that corresponds to the 32nd day of the month and is therefore invalid.</p>

Decode hourly shade rasters

Hourly shade data is encoded in multiband rasters. To learn more about raster basics, see [Solar API Concepts](#).

When you make a request for hourly shade data, you can receive up to 12 rasters, one for each month of the calendar year (January through December). Each raster is composed of 24 layers, or *bands*, which correspond to the 24 hours of the day.

Each band is represented by a matrix of cells, or *pixels*. Each pixel has a depth of 32 bits, which correspond to the (maximum) 31 days of the month. Decoding the day, time, and month of shade data, therefore, requires understanding the bit, band, and raster that you are analyzing.

For example, to identify whether a given location at coordinates (x, y) saw the sun at 4:00 PM on June 22, do the following:

1. Make a data layers request for all layers for location (x, y).
2. Because the month of June is the sixth month of the year, fetch the sixth URL in the `hourlyShadeUrls` list.
3. Hourly bands are given in 24-hour time. To get data for 4:00 PM (16:00), look up the 17th channel.
4. Bits (days) index from 0. To get data for the 22nd day of June, read bit 21.
5. Bits provide binary data indicating whether that location saw sun at the given date and time. If the bit is 1, the location saw sun. If the bit is 0, the location saw shade.

The following code summarizes the steps above:

```
(hourly_shade[month - 1])(x, y)[hour] & (1 << (day - 1))
```

Note: Hourly shade data assumes that there are no leap days and that Daylight Savings Time does not exist. All days are assumed to be 24 hours long and noon (when the sun is at its peak) is considered to be at "standard time."

[Previous](#)

[arrow_back](#) Make a data layers request

Was this helpful?

Recommended for you

Solar API Usage and Billing

The Solar API uses a pay-as-you-go pricing model. Solar API requests generate calls to one SKU for all but mobile-native apps. Along with the overall Google Terms of Use, there are usage limits specific to the Solar API. Manage your costs and usage

Updated Oct 5, 2023

Migrate to the Solar API on Google Maps Platform

To move from Google Earth Engine Solar API to the Google Maps Platform Solar API: Instructions here: Set up your Google Cloud project. Only certain roles can create a Cloud project; if you can't create a project, reach out to your organization's

Updated Oct 5, 2023

Compiling the Utilities

If the precompiled utilities don't suit your needs, you can build the WebP utilities yourself. Download libwebp-1.3.2.tar.gz from the downloads list and extract its contents. From the libwebp-1.3.2 directory, run: To see additional options, run: The

Updated Sep 14, 2023

Make a data layers request

 developers.google.com/maps/documentation/solar/data-layers

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

The `dataLayers` endpoint provides detailed solar information for a region surrounding a specified location. The endpoint returns 17 downloadable TIFF files, including:

- Digital surface model (DSM)
- RGB composite layer (aerial imagery)
- A mask layer that identifies the boundaries of the analysis
- Annual solar flux, or the annual yield of a given surface
- Monthly solar flux, or the monthly yield of a given surface
- Hourly shade (24 hours)

For more information about how the Solar API defines flux, see [Solar API Concepts](#).

About data layers requests

The following example shows the URL of a REST request to the `dataLayers` method:

```
https://solar.googleapis.com/v1/dataLayers:get?parameters
```

Include your request URL parameters that specify the following:

- Latitude and longitude coordinates of the location
- The radius of the region surrounding the location
- The subset of the data to return (DSM, RGB, mask, annual flux, or monthly flux)
- The minimum quality allowed in the results
- The minimum scale of data to return, in meters per pixels

Example data layers request

The following example requests all building insights information in a 100 meter radius for the location at the coordinates of latitude = 37.4450 and longitude = -122.1390:

```
curl -X GET "https://solar.googleapis.com/v1/dataLayers:get?  
location.latitude=37.4450&location.longitude=-122.1390&radiusMeters=100  
&view=FULL_LAYERS&requiredQuality=HIGH&pixelSizeMeters=0.5&key=YOUR_API_KEY"
```

The API returns URLs in the following format:

```
https://solar.googleapis.com/v1/solar/geoTiff:get?id=HASHED_ID
```

Example response

The request produces a JSON response in the form:


```
{
  "imageryDate": {
    "year": 2019,
    "month": 7,
    "day": 9
  },
  "imageryProcessedDate": {
    "year": 2022,
    "month": 3,
    "day": 21
  },
  "dsmUrl": "https://solar.googleapis.com/v1/geoTiff:get?
id=14f82e6931a8c33fc31ab8378e51804a-852f4ca7f056addda5b8fcb93e02c2fd",
  "rgbUrl": "https://solar.googleapis.com/v1/geoTiff:get?
id=bf769c43d72eb85493b20df583bc0c95-d13126638efaa89e44951abc8664d6a3",
  "maskUrl": "https://solar.googleapis.com/v1/geoTiff:get?
id=ed089240efc78e417c96a945460830ef-e666758b7cc183f82d1c7b7a891f858b",
  "annualFluxUrl": "https://solar.googleapis.com/v1/geoTiff:get?
id=aaa2637073d62cc7331d067eb7080bbe-f94eab79915f66759f5265b2ff8b1ad4",
  "monthlyFluxUrl": "https://solar.googleapis.com/v1/geoTiff:get?
id=d1608d342a3d0393b5decd063d330271-2a2e27504a2009cad1f1f3d2b471bcd3",
  "hourlyShadeUrls": [
    "https://solar.googleapis.com/v1/geoTiff:get?
id=541c2f32b936f190f7562309ea1d60fc-432bf94bcd0dc918f0c828d07aa00e7c",
    "https://solar.googleapis.com/v1/geoTiff:get?
id=4eb7a0b9c0f34e0e746816d0f3085274-4794b9eb35ab18ad4fbe2c3ee59f151d",
    ...
  ],
  "imageryQuality": "HIGH"
}
```

Access response data

Accessing data via response URLs requires additional authentication. If you use an authentication key, you must append your API key to the URL. If you use OAuth authentication, you must add OAuth headers.

To make a request to the URL in the response, append your API key to the URL:

```
https://solar.googleapis.com/v1/solar/geoTiff:get?
id=fbde33e9cd16d5fd10d19a19dc580bc1-
8614f599c5c264553f821cd034d5cf32&key=YOUR_API_KEY
```

Passing the API key provides you with better usage and analytics capabilities and better access control to the response data.

With the exception of the RGB layer, all TIFF files will display as blank images in image viewer applications. To view downloaded TIFF files, import them into a mapping application software, such as QGIS.

The full specification of this request and response is in the reference documentation.

Note: Response URLs are only active for one hour after the initial request. To access URLs beyond this timeframe, you must send a request to the dataLayers endpoint again.

Next

About GeoTIFF files [arrow_forward](#)

Was this helpful?

Recommended for you

Solar API Usage and Billing

The Solar API uses a pay-as-you-go pricing model. Solar API requests generate calls to one SKU for all but mobile-native apps. Along with the overall Google Terms of Use, there are usage limits specific to the Solar API. Manage your costs and usage

Updated Oct 5, 2023

About GeoTIFF files

The dataLayers endpoint returns data encoded as GeoTIFF files, which can be used in any geographic information system (GIS) application to design solar systems. Each string in the dataLayers response contains a URL, which you can use to fetch the

Updated Oct 5, 2023

Migrate to the Solar API on Google Maps Platform

To move from Google Earth Engine Solar API to the Google Maps Platform Solar API: Instructions here: Set up your Google Cloud project. Only certain roles can create a Cloud project; if you can't create a project, reach out to your organization's

Updated Oct 5, 2023

Calculate solar costs and savings for non-US locations

 developers.google.com/maps/documentation/solar/calculate-costs-non-us

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

This section describes how to do the calculations that enable you to determine the best solar configuration for households in non-US locations. To calculate the recommendations, you need to model the costs of installing solar panels and the savings that they provide by using the data from a Solar API response.

For US locations, the Solar API returns an instance of the `FinancialAnalysis` object for each electric bill size for the input location. You use the information in these instances to determine the bill, energy consumption, and, ultimately, the savings that are associated with each solar installation size.

For non-US locations, the API response doesn't include the `FinancialAnalysis` instances, so you have to calculate the cost and savings for each solar configuration yourself before you can recommend the best one. To perform the calculations, you need to gather location-specific data and follow the guidance in this document.

You can model your calculations on the calculations that the Solar API uses for US locations. For an explanation of these calculations, see [Calculate cost savings \(US\)](#).

Solar panel configurations

For non-US locations, the information about each solar panel configuration that you need for financial analysis is provided in the `SolarPanelConfig` field. The number of `SolarPanelConfig` instances that are returned depends on the roof size of the input location. For your calculations, you need the values from the following two fields:

- `panelsCount`: The number of panels that are used in this configuration.
- `yearlyEnergyDcKwh`: The amount of solar energy, in kWh of DC electricity, that this configuration produces over the course of a year, given the panel size defined by the following fields in the `SolarPotential` object:

The following example shows one instance of the `SolarPanelConfig` object in the `solarPanelConfigs` field in a request response:

```

"solarPanelConfigs": [
  {
    "panelsCount": 4,
    "yearlyEnergyDcKwh": 1709.2424,
    "roofSegmentSummaries": [
      {
        "pitchDegrees": 16.253168,
        "azimuthDegrees": 169.41516,
        "panelsCount": 4,
        "yearlyEnergyDcKwh": 1709.2424
      }
    ]
  }
]

```

For solar installations, `installationSize` refers to the kW output rather than the area or panel count and is defined as:

```
installationSize = panelsCount * panelCapacityWatts/1000 kW
```

Adjust energy production estimates for different panel ratings

To calculate the `yearlyEnergyDcKwh` value, the Solar API uses the power rating in the `panelCapacityWatts` field, which is currently 250W.

If you need to use a different panel power rating in your calculations and the dimensions of the panels are roughly comparable to the values in the `panelHeightMeters` and `panelWidthMeters` fields, you can adjust your calculations by multiplying the value returned by the API in the `yearlyEnergyDcKwh` field by the ratio of your power rating to the value in `panelCapacityWatts`.

For example, if the power rating of your panels is 400W and `panelCapacityWatts` is 250W, multiply the value of `yearlyEnergyDcKwh`, which the API calculated by using `panelCapacityWatts`, by a factor of 400/250, or 1.6. If your panel power rating is 200W, multiply `yearlyEnergyDcKwh` by 200/250, or 0.8.

Excess energy production

Accounting for excess energy that might be produced by a solar installation is out of scope for the Solar API calculations. In fact, if the Solar API returns multiple possible `SolarPanelConfig` instances for a given household, the Solar API doesn't consider results or configurations that produce more power than the assumed US average household consumption in the `FinancialAnalysis`.

However, you might have reasons for including installations that produce excess electricity in your recommendations. For example, you might want to offset the gradual decline in panel efficiency (the `efficiencyDepreciationFactor`) by allowing for excess production in the first part of an installation's life. For more information, see Required values for financial analysis.

Whatever your reasons, if you do include solar installations that produce excess electricity in your calculations, just be aware that the calculations that are explained here don't cover that scenario.

Required values for financial analysis for non-US locations

From each `SolarPanelConfig` instance in the API response, you need two values to perform the financial analysis for that instance:

- **panelsCount**: The number of solar panels in an installation. You use this value in your calculation of the `installationSize`.
- **yearlyEnergyDcKwh**: How much solar energy a layout captures over the course of a year, in kWh of DC electricity, given a specific `panelsCount`. You use this value in your calculation of the solar energy that is usable as AC electricity in a household (`initialAcKwhPerYear`) of each `installationSize`, taking into account any energy loss during the conversion from DC to AC.

Additionally, you need to gather location-specific values for the following variables that you will use in the calculations:

- **billCostModel()**: Your model for determining the cost, in local currency, paid by a household for using a given number of kWh. How much a utility charges for electricity can vary from day to day or hour to hour depending on things like demand, time of day, and how much electricity the household consumes. You might need to estimate an average cost.
- **costIncreaseFactor**: The factor by which the cost of electricity increases annually. The Solar API uses 1.022 (2.2% annual increase) for US locations. Adjust this value as needed for your area.
- **dcToAcDerate**: The efficiency at which an inverter converts the DC electricity that is produced by the solar panels to the AC electricity that is used in a household. The Solar API uses 85% for US locations. Adjust this value as needed for your area.
- **discountRate**: The Solar API uses 1.04 (4% annual increase) for US locations. Adjust this value as needed for your area.
- **efficiencyDepreciationFactor**: How much the efficiency of the solar panels declines each year. The Solar API uses 0.995 (0.5% annual decrease) for US locations. Adjust this value as needed for your area.
- **incentives**: Include any monetary incentives to install solar panels given by government entities in your area.
- **installationCostModel()**: Your method for estimating the cost of installing solar in local currency for a given `installationSize`. The cost model would typically account for local labor and material costs for a given `installationSize`.
- **installationLifeSpan**: The expected lifespan of the solar installation. The Solar API uses 20 years. Adjust this value as needed for your area.

- ***kWhConsumptionModel()***: Your model for determining how much energy a household consumes based on a monthly bill. In its simplest form, you would divide the bill by the average cost of a kWh in the household's location.
- ***monthlyBill***: the average monthly electric bill for a subject household.
- ***monthlyKWhEnergyConsumption***: An estimate of the average amount of electricity the household at a given location consumes in a month, measured in KWh.

With these values and the information provided by the API response, you can perform the calculations necessary to recommend the best **installationSize** for locations not covered by the Solar API.

Calculation steps

The following steps are based on the Solar API's methodology. You might need to adjust your methodology based on the information that is available for your location.

1. Calculate the annual energy consumption of the household at the input location:

1. Estimate or request the monthly bill for the household.
2. Calculate the *monthlyKWhEnergyConsumption* from the monthly bill. (If you know the *monthlyKWhEnergyConsumption*, you can skip this step.) For example:

```
| monthlyKWhEnergyConsumption = kWhConsumptionModel(monthlyBill)
```

1. Calculate *annualKWhEnergyConsumption* by multiplying *monthlyKWhEnergyConsumption* by 12:

```
| annualKWhEnergyConsumption = monthlyKWhEnergyConsumption x 12
```

2. Get the API response for the target household:

```
https://solar.googleapis.com/v1/buildingInsights:findClosest?
location.latitude=lat-number&location.longitude=long-number&key=yourAPIKey
```

The response includes usable sunlight, usable roof space, and one or more possible solar panel configurations.

3. Calculate the annual solar energy AC production of each **installationSize** the API proposes by multiplying the **yearlyEnergyDcKwh** value provided by the API in each **SolarPanelConfig** instance by your local *dcToAcDerate*:

```
| initialAcKwhPerYear = yearlyEnergyDcKwh x dcToAcDerate
```

4. Optionally, remove from consideration any **SolarPanelConfig** instance that produces more electricity than the household consumes annually (*initialAcKwhPerYear* > *annualKWhEnergyConsumption*).

5. Calculate the lifetime solar energy production (*LifetimeProductionAcKwh*) of each returned *installationSize*:

1. For each year of the lifespan of the solar installation, **calculate the amount of electricity the installation will produce annually**, applying the *efficiencyDepreciationFactor* exponentially to each year after the first.
2. Add the totals for all years.

The following table shows an example of how to calculate the lifetime energy production assuming an *installationLifeSpan* of 20 years. Each row represents a year of production. After the first year, the efficiency decline is applied exponentially. Finally, the sum of all rows is the lifetime energy production of the solar installation.

Year	Yearly solar energy production (kWh)
1	<i>initialAcKwhPerYear</i>
2	+ <i>initialAcKwhPerYear</i> x <i>efficiencyDepreciationFactor</i>
:	:
20	+ <i>initialAcKwhPerYear</i> x <i>efficiencyDepreciationFactor</i> ¹⁹
Total	<i>LifetimeProductionAcKwh</i>

Because the solar panel efficiency decays at a constant rate, it is essentially a geometric series where $a = \text{initialAcKwhPerYear}$ and $r = \text{efficiencyDepreciationFactor}$. We can use a geometric sum to calculate the **LifetimeProductionAcKwh**:

```
LifetimeProductionAcKwh = (dcToAcDerate * initialAcKwhPerYear * (1 -
pow(efficiencyDepreciationFactor, installationLifeSpan)) / (1 -
efficiencyDepreciationFactor))
```

The following Python code computes the geometric sum above:

```
def LifetimeProductionAckWh(
    dcToAcDerate,
    yearlyEnergyDcKwh,
    efficiencyDepreciationFactor,
    installationLifeSpan):
    return (
        dcToAcDerate *
        yearlyEnergyDcKwh *
        (1 - pow(
            efficiencyDepreciationFactor,
            installationLifeSpan)) /
        (1 - efficiencyDepreciationFactor))
```

1. For each returned **installationSize**, calculate the lifetime cost of energy consumption if the **installationSize** is installed:

1. For each year of the lifespan of the solar installation, **calculate the cost of the electricity the household will need to purchase annually to cover the energy consumption not met by solar power**. Use the values for *annualKWhEnergyConsumption* and *initialAcKwhPerYear* that you calculated previously. For each year after the first year, apply the *efficiencyDepreciationFactor*, *costIncreaseFactor*, and the *discountRate* to the values.
2. Add the totals for all years.

The following table shows an example of how to calculate the lifetime cost of electricity. Each row represents the cost of electricity for a year in the life of the solar installation. After the first year, both the increased cost of electricity and the discount rate are applied exponentially. Finally, the sum of all rows is the lifetime cost of electricity with the solar installation.

Annual utility bill in current local currency value (USD)	
Year	(<i>annualUtilityBillEstimate</i>)
1	<i>annualUtilityBillEstimateYear1 = billCostModel</i> <i>(yearlyKWhEnergyConsumption - initialAcKwhPerYear)</i>
2	<i>annualUtilityBillEstimateYear2 = billCostModel</i> <i>(yearlyKWhEnergyConsumption - initialAcKwhPerYear x</i> <i>efficiencyDepreciationFactor) x costIncreaseFactor / discountRate</i>
:	:
20	<i>annualUtilityBillEstimateYear20 = billCostModel</i> <i>(yearlyKWhEnergyConsumption - initialAcKwhPerYear x</i> <i>efficiencyDepreciationFactor19) x costIncreaseFactor19 / discountRate19</i>
Total	<i>remainingLifetimeUtilityBill</i>

The following Python code returns an array of `annualUtilityBillEstimate` for every year of the `installationLifeSpan`:

```
def annualUtilityBillEstimate(
    yearlyKWhEnergyConsumption,
    initialAckwhPerYear,
    efficiencyDepreciationFactor,
    year,
    costIncreaseFactor,
    discountRate):
    return (
        billCostModel(
            yearlyKWhEnergyConsumption -
            annualProduction(
                initialAckwhPerYear,
                efficiencyDepreciationFactor,
                year)) *
        pow(costIncreaseFactor, year) /
        pow(discountRate, year))
def lifetimeUtilityBill(
    yearlyKWhEnergyConsumption,
    initialAckwhPerYear,
    efficiencyDepreciationFactor,
    installationLifeSpan,
    costIncreaseFactor,
    discountRate):
    bill = [0] * installationLifeSpan
    for year in range(installationLifeSpan):
        bill[year] = annualUtilityBillEstimate(
            yearlyKWhEnergyConsumption,
            initialAckwhPerYear,
            efficiencyDepreciationFactor,
            year,
            costIncreaseFactor,
            discountRate)
    return bill
```

1. Calculate the lifetime cost of electricity if a solar installation is not installed:

1. For each year of the lifespan of the solar installation, **calculate the cost of the electricity the household will need to purchase annually if solar is not installed**. Use the value for *monthlyBill*. For each year after the first year, apply the *costIncreaseFactor* and the *discountRate* values to *monthlyBill*.

2. Add the totals for all years.

The following table shows an example of how to calculate the lifetime cost of electricity without solar. Each row represents the cost of electricity for a year over the same number of years as the lifespan of a solar installation. After the first year, both the increased cost of electricity and the discount rate are applied exponentially. Finally, the sum of all rows is the lifetime cost of electricity without solar installation.

Year	Annual utility bill in current local currency value
1	$annualBill = monthlyBill \times 12$
2	$annualBill = monthlyBill \times 12 \times costIncreaseFactor / discountRate$
:	:
20	$annualBill = monthlyBill \times 12 \times costIncreaseFactor^{19} / discountRate^{19}$
Total	<i>costOfElectricityWithoutSolar</i>

The following code performs the calculation above:

```
lifetimeBill = (
    monthlyBill * 12 *
    (1 - pow(costIncreaseFactor / discountRate, installationLifeSpan)) /
    (1 - costIncreaseFactor / discountRate))
```

1. For each installation size, calculate the installation cost:

$installationCost = localInstallationCostModel(installationSize)$

2. Add up any monetary incentives that are available to the household location.

3. For each installation size, calculate the total costs associated with installing solar:

$totalCostWithSolar = installationCost + remainingLifetimeUtilityBill - incentives$

4. For each installation size, calculate the total savings associated with installing solar:

$savings = costOfElectricityWithoutSolar - totalCostWithSolar$

5. Select the installation size that provides the most savings.

When your calculations are done

Using the information you provide, the information returned by the Solar API, and the above calculations, you should be able to recommend solar installation sizes that provide maximum cost savings for households in your area.

In the recommendations that you provide to your end user, you can also include the following information that is returned by the API in the `SolarPotential` object of the `solarPotential` field:

- How much usable sunlight a house receives annually, which is returned in the `maxSunshineHoursPerYear` field of the `SolarPotential` object.
- How many square feet of a roof can be used for a solar installation, which is returned in the `wholeRoofStats` field of the `SolarPotential` object.
- The average monthly electricity bill for the household.

[Previous](#)

[arrow_back](#) Calculate costs and savings (US only)

Was this helpful?

Recommended for you

Solar API Usage and Billing

The Solar API uses a pay-as-you-go pricing model. Solar API requests generate calls to one SKU for all but mobile-native apps. Along with the overall Google Terms of Use, there are usage limits specific to the Solar API. Manage your costs and usage

Updated Oct 5, 2023

Make a data layers request

The dataLayers endpoint provides detailed solar information for a region surrounding a specified location. The endpoint returns 17 downloadable TIFF files, including: For more information about how the Solar API defines flux, see Solar API Concepts.

Updated Oct 5, 2023

About GeoTIFF files

The dataLayers endpoint returns data encoded as GeoTIFF files, which can be used in any geographic information system (GIS) application to design solar systems. Each string in the dataLayers response contains a URL, which you can use to fetch the

Updated Oct 5, 2023

Calculate solar costs and savings (US only)

 developers.google.com/maps/documentation/solar/calculate-costs-us

This document explains how the Solar API calculates the various values that it uses to recommend solar panel installations and to estimate the costs and cost savings for US addresses.

If you enter the address of a residence in a covered region of the US, the Solar API shows you the following estimates:

- How much sunlight the house receives annually
- How much space the roof has for a solar installation
- How much savings, in US dollars, the home can expect over the 20 year life of a solar system
- The average monthly electricity bill for homes in your area, which you can adjust for your home
- A recommended size, measured in kilowatts (kW), for a solar system on the house

Although the Solar API provides estimates for any structure that it has data for, the estimates that it provides are best suited for residences or small commercial structures. The Solar API recommends solar installation sizes that maximize savings without producing more energy in a year than a household can consume. The Solar API does not calculate values related to excess energy production.

The recommended installation sizes are limited to annual energy consumption for a number of reasons, but primarily because US households currently get little or no financial benefit from excess energy production. In US locations that have net metering, credits earned from excess energy production typically expire over time.

Required values for financial analysis for non-US locations

From each `SolarPanelConfig` instance in the API response, you need two values to perform the financial analysis for that instance:

- **panelsCount**: The number of solar panels in an installation. You use this value in your calculation of the `installationSize`.
- **yearlyEnergyDcKwh**: How much sunlight energy a layout captures over the course of a year, in DC kWh, given a specific `panelsCount`. You use this value in your calculation of the annual solar energy AC production (`initialAcKwhPerYear`) of each `installationSize`.

Additionally, you need to gather location-specific values for the following variables that you will use in the calculations:

- **billCostModel()**: Your model for determining the cost, in local currency, paid by a household for using a given number of kWh. How much a utility charges for electricity can vary from day to day or hour to hour depending on things like demand, time of day, and how much electricity the household consumes. You might need to estimate an average cost.
- **costIncreaseFactor**: The Solar API uses 1.022 (2.2% annual increase) for US locations.
- **dcToAcDerate**: The efficiency at which an inverter converts the DC electricity that is produced by the solar panels to the AC electricity that is used in a household. The Solar API uses 85% for US locations.
- **discountRate**: The Solar API uses 1.04 (4% annual increase) for US locations.
- **efficiencyDepreciationFactor**: How much the efficiency of the solar panels declines each year. The Solar API uses 0.995 (0.5% annual decrease) for US locations.
- **incentives**: Include any monetary incentives to install solar panels given by government entities in your area.
- **installationCostModel()**: Your method for estimating the cost of installing solar in local currency for a given **installationSize**. The cost model would typically account for local labor and material costs for a given **installationSize**.
- **installationLifeSpan**: The expected lifespan of the solar installation. The Solar API uses 20 years. Adjust this value as needed for your area.
- **kWhConsumptionModel()**: Your model for determining how much energy a household consumes based on a monthly bill. In its simplest form, you would divide the bill by the average cost of a kWh in the household's location.
- **monthlyBill**: the average monthly electric bill for a subject household.
- **monthlyKWhEnergyConsumption**: An estimate of the average amount of electricity the household at a given location consumes in a month, measured in kWh.

With these values and the information provided by the API response, you can perform the calculations necessary to recommend the best **installationSize** for locations not covered by the Solar API.

How it works

The average monthly electric bill is the key to the rest of the calculations.

The Solar API initially bases its calculations on a preselected monthly bill amount. If needed, you can select a different amount that more accurately reflects your own average monthly bill.

Knowing the amount of a monthly bill and the current cost of electricity in a given location, the Solar API can estimate the number of kilowatt hours (kWh) of electricity a household consumes each month. For the current cost of electricity around the US, and to determine

kWhs from a monthly bill, the Solar API references databases maintained by Clean Power Research.

Using the number of kWh a household consumes, the usable area of a home's roof, and the solar potential of the home's location, the Solar API evaluates one or more possible solar installation sizes and recommends the size that provides the most savings.

The size of a solar panel installation is measured by its kW rating. The kW rating depends on the number of solar panels in the configuration and the power rating, measured in watts, of each panel.

The kW rating of an installation is not the same as the energy output of an installation, which is measured in kWh and is variable. The kWh output of an installation is dependent on factors like the following:

- The time of day
- The weather
- The orientation of the panel to the sun
- Any shadows cast on the panels by nearby objects
- The regional solar potential
- The age of the installation

The Solar API includes factors like regional solar potential and the age of the installation in its estimate of the annual energy production of a solar installation.

To determine the usable area of a roof and estimate the solar installation size it can support, the Solar API uses aerial imagery and advanced 3D modelling.

Detailed explanation of the values and calculations

The following sections explain how the Solar API calculates the costs, savings, and sizes of solar installations for a given structure in the U.S.

The explanations of the calculations use terms to represent values in the calculations. For an explanation of the terms, see Definition of the terms used in our calculations.

Annual household energy consumption

As mentioned earlier, the Solar API determines the monthly consumption of electricity based on the monthly bill amount and the cost of electricity where a household is located. After determining the monthly consumption of electricity of a household, we calculate annual energy consumption in kWh by using the following formula:

```
annualKWhEnergyConsumption = monthlyKWhEnergyConsumption x 12
```

A household's energy consumption is assumed to remain the same year to year over the life of a solar installation. The Solar API assumes the life of a solar installation to be 20 years.

Annual solar energy production

The Solar API estimates the annual energy production of a solar installation by considering factors like the intensity of sunlight, angle of sunlight, and number of hours of usable sunlight a region gets annually.

Solar installations produce direct current (DC) electricity, which has to be converted to alternating current (AC) electricity by an inverter before you can use it in your home. Some electricity is lost during the conversion process, and the efficiency of the inverter determines how much is lost.

The efficiency of the conversion process is referred to as the *DC to AC derate*. To account for the loss, the Solar API multiplies the annual output of the solar installation by a DC to AC derate of 0.85. The result is the annual production of AC electricity, as shown in the following formula:

$$\text{initialAcKwhPerYear} = \text{yearlyEnergyDcKwh} \times 0.85$$

The amount of energy an installation produces declines by about 0.5% each year over the life of the installation. To account for this, after the first year, the Solar API multiplies the annual AC output of an installation by 99.5%, or 0.995, each year over the estimated 20-year lifetime of the installation. This is illustrated in the following table.

Year Yearly solar energy production (kWh)

1	<i>initialAcKwhPerYear</i>
2	<i>initialAcKwhPerYear</i> x 0.995
:	:
20	<i>initialAcKwhPerYear</i> x 0.995 ¹⁹

Because the solar panel efficiency decays at a constant rate, it is essentially a geometric series where $a = \text{initialAcKwhPerYear}$ and $r = \text{efficiencyDepreciationFactor}$. We can use a geometric sum to calculate the **LifetimeProductionAcKwh**:

$$\text{LifetimeProductionAcKwh} = (\text{dcToAcDerate} * \text{initialAcKwhPerYear} * (1 - \text{pow}(\text{efficiencyDepreciationFactor}, \text{installationLifeSpan})) / (1 - \text{efficiencyDepreciationFactor}))$$

The cost of electricity with solar If the size of an installation is limited by the roof size or other factors, the solar installation might produce less electricity than a household consumes. In these cases, the household will likely have to pay a utility for some amount

of electricity each year, as shown in the following formula:

`annualKWhEnergyConsumption - initialAcKwhPerYear = annualUtilityEnergyRequired`
To account for this cost, the Solar API applies a bill cost model to the estimated amount of electricity, in kWh, the household will need from a utility over the life of the solar installation. The following formula illustrates this calculation:

`annualUtilityBillEstimate = billCostModel(utilityEnergyRequired)`

To account for the yearly increase in the cost of electricity, we apply a *costIncreaseFactor* of 2.2%, or 0.022, per year for US locations:

`costIncreaseFactor = 1 + 2.2% = 1.022`

Due to inflation, we have to discount the value of the currency value in our estimates of future costs. To account for this, we apply a 4% discount rate to our model for US locations:

`discountRate = 1 + 4% = 1.04`

The following table shows how each year's utility bill is calculated over the life of a solar installation. The *remainingLifetimeUtilityBill* is the sum total of the utility bills for each of the 20 years of the solar installation's lifetime.

Annual utility bill in current local currency value (USD)	
Year	(<i>annualUtilityBillEstimate</i>)
1	<i>billCostModel (yearlyKWhEnergyConsumption - initialAcKwhPerYear) = annualUtilityBillEstimateYear1</i>
2	<i>billCostModel (yearlyKWhEnergyConsumption - initialAcKwhPerYear x 0.995) x 1.022 / 1.04 = annualUtilityBillEstimateYear2</i>
:	:
20	<i>billCostModel (yearlyKWhEnergyConsumption - initialAcKwhPerYear x 0.995¹⁹) x 1.022¹⁹ / 1.04¹⁹ = annualUtilityBillEstimateYear20</i>
<i>remainingLifetimeUtilityBill = annualUtilityBillEstimateYear1 +</i>	
Total	<i>annualUtilityBillEstimateYear2 + + annualUtilityBillEstimateYear20</i>

The cost of electricity without solar

To calculate how much a household might save if they install solar, we also have to calculate how much the household might pay if they don't.

We again have to account for the increasing cost of electricity and inflation by applying the *costIncreaseFactor* of 1.022 and the *discountRate* of 1.04 to the calculation, as we did when we calculated the cost of electricity with solar.

The following table shows how each year's utility bill without solar is calculated over the life of a solar installation. The *costOfElectricityWithoutSolar* is the sum total of the utility bills over the same 20 year period that we used for the cost of electricity with solar.

Year	Yearly utility bill (USD)
1	$monthlyBill \times 12$
2	$monthlyBill \times 12 \times 1.022 / 1.04$
:	:
20	$monthlyBill \times 12 \times 1.022^{19} / 1.04^{19}$
Sum of all annual bills, which can also be expressed as	
Total	$costOfElectricityWithoutSolar = 204.35 \times monthlyBill$

The cost of installing solar

The Solar API includes the cost of installing the recommended solar configuration in the estimates that it provides. To estimate the cost of an installation, the Solar API uses a localized installation cost model and the size of the installation.

```
installationCost = InstallationCostModel (installationSize)
```

Incentives

Government entities might provide incentives for installing solar. The incentives are often in the form of tax credits. Based on a household's location, the Solar API subtracts any incentives that are currently available to the household from the estimate of the total costs.

The total cost with solar installation

The Solar API calculates the total 20-year cost of a solar configuration by using the following formula:

```
totalCostWithSolar = installationCost + remainingLifetimeUtilityBill - incentives
```

The total savings

The Solar API calculates the savings for the household using the following formula:

```
savings = costOfElectricityWithoutSolar - totalCostWithSolar
```

The Solar API performs the above calculations for each possible installation size and then recommends the installation size that provides the maximum savings for the household. The amount of the estimated savings is returned with the recommendation.

Make a building insights request

 developers.google.com/maps/documentation/solar/building-insights

The buildingInsights endpoint provides insights about the location, dimensions, and solar potential of a building. In particular, you can get information about:

- Solar potential, including solar panel size, annual amount of sunshine, carbon offset factors, and more
- Solar panel position, orientation, and energy production
- Estimated monthly energy bill of optimal solar layout and associated costs and benefits

To learn more about how the Solar API defines solar potential and sunniness, see Solar API Concepts.

About building insights requests

To request building insights, send an HTTP GET request to:

```
https://solar.googleapis.com/v1/buildingInsights:findClosest?key=YOUR_API_KEY
```

Include your request URL parameters that specify the latitude and longitude coordinates of the location and the minimum required quality level allowed in the results.

Example building insights request

The following example requests building insights information for the location at the coordinates of latitude = 37.4450 and longitude = -122.1390:

```
https://solar.googleapis.com/v1/buildingInsights:findClosest?  
location.latitude=37.4450&location.longitude=-122.1390&requiredQuality=HIGH  
&key=YOUR_API_KEY
```

That request produces a JSON response in the form:

```
{
  "name": "buildings/ChIJh0CMPQW7j4ARLrRiVvmg6Vs",
  "center": {
    "latitude": 37.4449439,
    "longitude": -122.13911639999999
  },
  "imageryDate": {
    "year": 2018,
    "month": 6,
    "day": 4
  },
  "postalCode": "94303",
  "administrativeArea": "CA",
  "statisticalArea": "06085511100",
  "regionCode": "US",
  "solarPotential": {
    "maxArrayPanelsCount": 1373,
    "maxArrayAreaMeters2": 2247.3264,
    "maxSunshineHoursPerYear": 1809.6869,
    "carbonOffsetFactorKgPerMwh": 428.9201,
    "wholeRoofStats": {
      "areaMeters2": 2861.0686,
      "sunshineQuantiles": [
        384.4651,
        1385.7468,
        1465.545,
        1523.7301,
        1553.636,
        1589.27,
        1619.2816,
        1640.2871,
        1663.76,
        1750.4572,
        1883.4658
      ],
      "groundAreaMeters2": 2740.45
    },
    "roofSegmentStats": [
      {
        "pitchDegrees": 11.54571,
        "azimuthDegrees": 269.49054,
        "stats": {
          "areaMeters2": 478.75748,
          "sunshineQuantiles": [
            477.1805,
            1391.751,
            1515.6603,
            1555.4863,
            1579.1082,
            1594.9426,
            1607.4075,
            1621.5597,
            1631.4178,
            1641.8871,
            1727.1367
          ]
        }
      }
    ]
  }
}
```

```

    ],
    "groundAreaMeters2": 469.07
  },
  "center": {
    "latitude": 37.4449647,
    "longitude": -122.1393672
  },
  "boundingBox": {
    "sw": {
      "latitude": 37.4447303,
      "longitude": -122.1394371
    },
    "ne": {
      "latitude": 37.4451819,
      "longitude": -122.1392963
    }
  },
  "planeHeightAtCenterMeters": 8.942595
},
],
"solarPanelConfigs": [
  {
    "panelsCount": 4,
    "yearlyEnergyDcKwh": 1823.9904,
    "roofSegmentSummaries": [
      {
        "pitchDegrees": 12.41514,
        "azimuthDegrees": 179.66463,
        "panelsCount": 4,
        "yearlyEnergyDcKwh": 1823.9905,
        "segmentIndex": 2
      }
    ]
  },
  /.../
]
"financialAnalyses": [
  {
    "monthlyBill": {
      "currencyCode": "USD",
      "units": "20"
    },
    "panelConfigIndex": -1
  },
  {
    "monthlyBill": {
      "currencyCode": "USD",
      "units": "25"
    },
    "panelConfigIndex": -1
  },
  {
    "monthlyBill": {
      "currencyCode": "USD",
      "units": "30"
    }
  }
]

```

```
    },  
    "panelConfigIndex": -1  
  },  
  {  
    "monthlyBill": {  
      "currencyCode": "USD",  
      "units": "35"  
    },  
    "panelConfigIndex": 0,  
    "financialDetails": {  
      "initialAckWhPerYear": 1550.3918,  
      "remainingLifetimeUtilityBill": {  
        "currencyCode": "USD",  
        "units": "2548"  
      },  
      "federalIncentive": {  
        "currencyCode": "USD",  
        "units": "1483"  
      },  
      "stateIncentive": {  
        "currencyCode": "USD"  
      },  
      "utilityIncentive": {  
        "currencyCode": "USD"  
      },  
      "lifetimeSrecTotal": {  
        "currencyCode": "USD"  
      },  
      "costOfElectricityWithoutSolar": {  
        "currencyCode": "USD",  
        "units": "10362"  
      },  
      "netMeteringAllowed": true,  
      "solarPercentage": 86.94348,  
      "percentageExportedToGrid": 52.350193  
    },  
    "leasingSavings": {  
      "leasesAllowed": true,  
      "leasesSupported": true,  
      "annualLeasingCost": {  
        "currencyCode": "USD",  
        "units": "335",  
        "nanos": 85540771  
      },  
      "savings": {  
        "savingsYear1": {  
          "currencyCode": "USD",  
          "units": "-9"  
        },  
        "savingsYear20": {  
          "currencyCode": "USD",  
          "units": "1113"  
        },  
        "presentValueOfSavingsYear20": {  
          "currencyCode": "USD",
```

```
      "units": "579",
      "nanos": 113281250
    },
    "financiallyViable": true,
    "savingsLifetime": {
      "currencyCode": "USD",
      "units": "1113"
    },
    "presentValueOfSavingsLifetime": {
      "currencyCode": "USD",
      "units": "579",
      "nanos": 113281250
    }
  }
},
"cashPurchaseSavings": {
  "outOfPocketCost": {
    "currencyCode": "USD",
    "units": "5704"
  },
  "upfrontCost": {
    "currencyCode": "USD",
    "units": "4221"
  },
  "rebateValue": {
    "currencyCode": "USD",
    "units": "1483",
    "nanos": 40039063
  },
  "paybackYears": 11.5,
  "savings": {
    "savingsYear1": {
      "currencyCode": "USD",
      "units": "326"
    },
    "savingsYear20": {
      "currencyCode": "USD",
      "units": "7815"
    },
    "presentValueOfSavingsYear20": {
      "currencyCode": "USD",
      "units": "1094",
      "nanos": 233276367
    },
    "financiallyViable": true,
    "savingsLifetime": {
      "currencyCode": "USD",
      "units": "7815"
    },
    "presentValueOfSavingsLifetime": {
      "currencyCode": "USD",
      "units": "1094",
      "nanos": 233276367
    }
  }
}
```



```

    },
    "financedPurchaseSavings": {
      "annualLoanPayment": {
        "currencyCode": "USD",
        "units": "335",
        "nanos": 85540771
      },
      "rebateValue": {
        "currencyCode": "USD"
      },
    },
    "loanInterestRate": 0.05,
    "savings": {
      "savingsYear1": {
        "currencyCode": "USD",
        "units": "-9"
      },
      "savingsYear20": {
        "currencyCode": "USD",
        "units": "1113"
      },
      "presentValueOfSavingsYear20": {
        "currencyCode": "USD",
        "units": "579",
        "nanos": 113281250
      },
      "financiallyViable": true,
      "savingsLifetime": {
        "currencyCode": "USD",
        "units": "1113"
      },
      "presentValueOfSavingsLifetime": {
        "currencyCode": "USD",
        "units": "579",
        "nanos": 113281250
      }
    }
  },
  ],
  /.../
  "panelCapacityWatts": 250,
  "panelHeightMeters": 1.65,
  "panelWidthMeters": 0.992,
  "panelLifetimeYears": 20,
  "buildingStats": {
    "areaMeters2": 2945.2869,
    "sunshineQuantiles": [
      372.5415,
      1371.2333,
      1456.3909,
      1519.6279,
      1550.9833,
      1586.4949,
      1617.6183,
      1639.3303,

```

```
        1662.812,  
        1748.7468,  
        1892.1855  
    ],  
    "groundAreaMeters2": 2789.4  
},  
"solarPanels": [  
  {  
    "center": {  
      "latitude": 37.4449709,  
      "longitude": -122.13907649999999  
    },  
    "orientation": "LANDSCAPE",  
    "yearlyEnergyDcKwh": 456.5196,  
    "segmentIndex": 2  
  },  
  /.../  
]  
"imageryQuality": "HIGH",  
"imageryProcessedDate": {  
  "year": 2022,  
  "month": 10,  
  "day": 16  
}  
}
```

Use OAuth | Solar API | Google for Developers

 developers.google.com/maps/documentation/solar/oauth-token

- [Home](#)
- [Products](#)
- [Google Maps Platform](#)
- [Documentation](#)
- [Solar API](#)

Was this helpful?

Use OAuth

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

Solar API supports the use of OAuth 2.0 for authentication. Google supports common OAuth 2.0 scenarios such as those for a web server.

This document describes how to pass an OAuth token to the Solar API call in your **development** environment. For instructions on using OAuth in a **production** environment, see [Authentication at Google](#).

Before you begin

Before you start using the Solar API, you need a project with a billing account and the Solar API enabled. We recommend creating multiple Project Owners and Billing Administrators, so that you'll always have someone with these roles available to your team. To learn more, see [Set up in Cloud Console](#).

About OAuth

There are many ways to create and manage access tokens with OAuth based on your deployment environment.

For example, the Google OAuth 2.0 system supports server-to-server interactions, such as those between your application and a Google service. For this scenario you need a service account, which is an account that belongs to your application instead of to an individual end user. Your application calls Google APIs on behalf of the service account, so users aren't directly involved. For more information on authentication methods, see [Authentication at Google](#).

Alternatively, you might use the Solar API as part of an Android or iOS mobile app. For general information on using OAuth with the Solar API, including information on managing access tokens for different deployment environments, see [Using OAuth 2.0 to Access Google APIs](#).

About OAuth scopes

To use OAuth with the Solar API, the OAuth token must be assigned the scope:

<https://www.googleapis.com/auth/cloud-platform>

Example: Try REST API calls in your local development environment

If you want to try the Solar API using an OAuth token, but do not have an environment setup to generate tokens, you can use the procedure in this section to make the call.

This example describes how to use the OAuth token provided by Application Default Credentials (ADC) to make the call. For information about using ADC to call Google APIs using client libraries, see [Authenticate using client libraries](#).

Note: The procedure below to make a REST call is not intended for use in a production environment. Use this procedure for a development or testing environment only.

Prerequisites

Before you can make a REST request using ADC, use the Google Cloud CLI to provide credentials to ADC:

1. If you haven't already, create a project and enable billing by following the steps in the [Set Up](#) in the Google Cloud Console.
2. Install and initialize the gcloud CLI.
3. Run the following `gcloud` command on your local machine to create your credential file:

```
gcloud auth application-default login
```

4. A login screen is displayed. After you log in, your credentials are stored in the local credential file used by ADC.

For more information, see [Local development environment](#) section of the [Provide credentials for Application Default Credentials](#) documentation.

Make a REST request

In this example, you pass two request headers:

- Pass the OAuth token in the **Authorization** header by using the following command to generate the token:

```
gcloud auth application-default print-access-token
```

The returned token has a scope of <https://www.googleapis.com/auth/cloud-platform>.

- Pass the ID or name of your Google Cloud project that has billing enabled in the **X-Goog-User-Project** header. To learn more, see [Set up in Cloud Console](#).

The following example makes a call to the Solar API using an OAuth token:

```
curl -X GET \  
-H 'X-Goog-User-Project: PROJECT_NUMBER_OR_ID' \  
-H "Authorization: Bearer $TOKEN" \  
"https://solar.googleapis.com/v1/dataLayers:get?location.latitude=37.4450  
&location.longitude=-122.1390&radius_meters=100&required_quality=HIGH"
```

Troubleshooting

If your request returns an error message about end-user credentials not being supported by this API, see [User credentials not working](#).

Was this helpful?

Recommended for you

Solar API Usage and Billing

The Solar API uses a pay-as-you-go pricing model. Solar API requests generate calls to one SKU for all but mobile-native apps. Along with the overall Google Terms of Use, there are usage limits specific to the Solar API. Manage your costs and usage

Updated Oct 5, 2023

Make a building insights request

The `buildingInsights` endpoint provides insights about the location, dimensions, and solar potential of a building. In particular, you can get information about: To learn more about how the Solar API defines solar potential and sunniness, see [Solar](#)

Updated Oct 5, 2023

Make a data layers request

The `dataLayers` endpoint provides detailed solar information for a region surrounding a specified location. The endpoint returns 17 downloadable TIFF files, including: For more information about how the Solar API defines flux, see [Solar API Concepts](#).

Updated Oct 5, 2023

Use API Keys with Solar API

 developers.google.com/maps/documentation/solar/get-api-key

- [Home](#)
- [Products](#)
- [Google Maps Platform](#)
- [Documentation](#)
- [Solar API](#)

Was this helpful?

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

Google Maps Platform products are secured from unauthorized use by restricting API calls to those that provide proper authentication credentials. These credentials are in the form of an API key - a unique alphanumeric string that associates your Google billing account with your project, and with the specific API or SDK.

This guide shows how to create, restrict, and use your API key for Google Maps Platform.

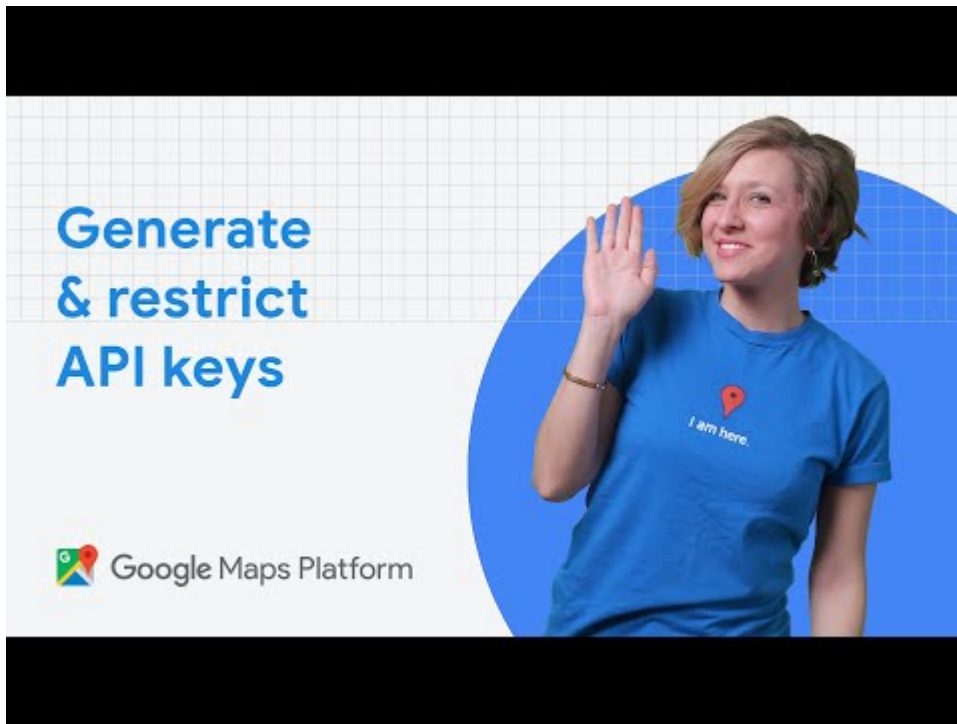
Before you begin: Before you start using the Solar API, you need a project with a billing account and the Solar API enabled. We recommend creating multiple Project Owners and Billing Administrators, so that you'll always have someone with these roles available to your team. To learn more, see [Set up in Cloud Console](#).

Creating API keys

The API key is a unique identifier that authenticates requests associated with your project for usage and billing purposes. You must have at least one API key associated with your project.

To create an API key:

Console Cloud SDK



Watch Video At: https://youtu.be/2_HZObVbe-g

1. Go to the **Google Maps Platform > Credentials** page.

Go to the Credentials page

2. On the **Credentials** page, click **Create credentials > API key**.
The **API key created** dialog displays your newly created API key.
3. Click **Close**.
The new API key is listed on the **Credentials** page under **API keys**.
(Remember to restrict the API key before using it in production.)

Restricting API keys

Google strongly recommends that you restrict your API keys by limiting their usage to those only APIs needed for your application. Restricting API keys adds security to your application by protecting it from unwarranted requests. For more information, see API security best practices.

When restricting an API key in the Cloud Console, **Application restrictions** override any APIs enabled under **API restrictions**. Follow best practices by creating a separate API key for each app, and for each platform on which that app is available.

To restrict an API key:

ConsoleCloud SDK

1. Go to the **Google Maps Platform > Credentials** page.

Go to the Credentials page

2. Select the API key that you want to set a restriction on. The API key property page appears.
3. Under **Key restrictions**, set the following restrictions:
 - Application restrictions:

To accept requests from the list of web server IP addresses that you supply, select **IP addresses (web servers, cron jobs, etc.)** from the list of **Application restrictions**. Specify one or more IPv4 or IPv6 address, or subnet using CIDR notation. The IP addresses must match the source address the Google Maps Platform servers observe. If you use network address translation (NAT), this would typically correspond to your machine's *public* IP address.
 - API restrictions:
 1. Click **Restrict key**.
 2. Select **Solar API** from **Select APIs** dropdown. If the Solar API is not listed, you need to enable it.
4. To finalize your changes, click **Save**.

Adding the API key to your request

You must include an API key with every Solar API request. In the following example, replace **YOUR_API_KEY** with your API key.

```
https://solar.googleapis.com/v1/buildingInsights:findClosest?
location.latitude=37.2746464&location.longitude=-121.7530949&key=YOUR_API_KEY
```

HTTPS is required for requests that use an API key.

Note: When using Web Services, all special characters inside the parameters must be URL encoded. URLs must be properly encoded to be valid and are limited to 2048 characters for all web services. Be aware of this limit when constructing your URLs. Different browsers, proxies, and servers may have different URL character limits as well. Was this helpful?

Recommended for you

Solar API Usage and Billing

The Solar API uses a pay-as-you-go pricing model. Solar API requests generate calls to one SKU for all but mobile-native apps. Along with the overall Google Terms of Use, there are usage limits specific to the Solar API. Manage your costs and usage

Updated Oct 5, 2023

Make a building insights request

The buildingInsights endpoint provides insights about the location, dimensions, and solar potential of a building. In particular, you can get information about: To learn more about how the Solar API defines solar potential and sunniness, see Solar

Updated Oct 5, 2023

Make a data layers request

The dataLayers endpoint provides detailed solar information for a region surrounding a specified location. The endpoint returns 17 downloadable TIFF files, including: For more information about how the Solar API defines flux, see Solar API Concepts.

Updated Oct 5, 2023

Set up your Google Cloud project

 developers.google.com/maps/documentation/solar/cloud-setup

- [Home](#)
- [Products](#)
- [Google Maps Platform](#)
- [Documentation](#)
- [Solar API](#)

Was this helpful?

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

This guide shows how to set up your Google Cloud project before using the Google Maps Platform APIs. While you may have completed some of these steps in the Getting started with Google Maps Platform page, this topic provides additional, useful instructions for managing your projects.

Create a project

To use Google Maps Platform, you must have a project to manage services, credentials, billing, APIs, and SDKs.

Billing setup is required for each project, but you will only be charged if a project exceeds its quota of no-charge services.

To create a Cloud project with billing enabled:

Consolegcloud

1. Create a new Google Cloud project in the Cloud Console:

Create new project

2. On the **New Project** page, fill in the required information:

- **Project name:** Accept the default or enter a customized name.

You can change the project name at any time. For more information, see [Identifying projects](#).

- **Project ID:** Accept the default or click **EDIT** to enter a customized ID that Google APIs use as a unique identifier for your project.

After you create the project, you *cannot* change the project ID, so choose an ID that you'll be comfortable using for the lifetime of the project. Don't include any sensitive information in your project ID.

- **Billing account:** Select a billing account for the project. If you haven't set up a billing account or only have one billing account, you won't see this option.

You must be a Billing Account Administrator or Project Billing Manager to associate a project with a billing account. For more information, see the [billing access control documentation](#).

- **Location:** If you have an organization you want to link your project to, click **Browse** and select it; otherwise, choose "No organization".

For more information, see [Creating and Managing Folders and Relationships between organizations, projects, and billing accounts](#).

3. Select **Create**.

Enable billing

To deploy your apps, you must enable billing. Your account will not be charged if you stay within your monthly quota. If your application needs resources that exceed the monthly quota, you will be charged for the additional usage.

If you have a billing account when you create a Cloud project, then billing is automatically enabled on that project.

To enable billing on a Cloud project:

1. In the Cloud Console, go to the Billing page:
Go to the Billing page
2. Select or create a Cloud project.

3. Depending on if a billing account exists or if the selected Cloud project is associated with an account, the Billing page displays one of the following:

- If billing is already enabled for the selected Cloud project, then the details about the billing account are listed.
- If no billing account exists, you are prompted to create a billing account and associate it with the selected Cloud project.
- If a billing account exists, you are prompted to enable billing if the selected Cloud project is not already associated with a billing account. You can also select **Cancel** and then select **Create account** to create and associate a new billing account.

After you enable billing, there is no limit to the amount that you might be charged. To gain more control over your costs, you can create a budget and set alerts. For more information, see [Billing](#).

Enable APIs

To use Google Maps Platform, you must enable the APIs or SDKs you plan to use with your project.

View enabled APIs

Consolegcloud

To view the APIs or SDKs you have enabled, go to the Google Maps Platform page in the Cloud Console:

Go to Google Maps Platform page

- Additional APIs: These API or SDKs are not enabled.
- If you see cards for each Map APIs and services, no APIs or SDKs have been enabled.

Shut down a project

You can disable billing and release all the Cloud resources that are being used in your Cloud project by shutting down that project:

Consolegcloud

1. Go to the Projects page:

Go to the Projects page

2. Select the Cloud project that you want to shut down, then click **Delete**.

For more information about managing your Cloud projects, see [Cloud Resource Manager](#): creating, shutting down, and restoring projects.

What's next

After setting up your Google Cloud project, you must create and secure your API Key to use the Solar API:

Use API Keys

Or, you can create an OAuth token:

Use OAuth

Was this helpful?

Recommended for you

Solar API Usage and Billing

The Solar API uses a pay-as-you-go pricing model. Solar API requests generate calls to one SKU for all but mobile-native apps. Along with the overall Google Terms of Use, there are usage limits specific to the Solar API. Manage your costs and usage

Updated Oct 5, 2023

Make a building insights request

The buildingInsights endpoint provides insights about the location, dimensions, and solar potential of a building. In particular, you can get information about: To learn more about how the Solar API defines solar potential and sunniness, see Solar

Updated Oct 5, 2023

Make a data layers request

The dataLayers endpoint provides detailed solar information for a region surrounding a specified location. The endpoint returns 17 downloadable TIFF files, including: For more information about how the Solar API defines flux, see Solar API Concepts.

Updated Oct 5, 2023

Solar API coverage

 developers.google.com/maps/documentation/solar/coverage

- [Home](#)
- [Products](#)
- [Google Maps Platform](#)
- [Documentation](#)
- [Solar API](#)

Was this helpful?

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

The Solar API provides solar data for hundreds of millions of buildings across the world. Imagery quality can be **HIGH**, **MEDIUM**, or **LOW**.

- **HIGH:** Solar data was based on high-resolution (e.g., 10cm) DSM data, typically from low-altitude aerial imagery.
- **MEDIUM:** Solar data was based on medium-resolution (e.g., 25cm) DSM data, typically from high-altitude aerial imagery.
- **LOW:** Solar data was based on low-resolution (e.g., 50cm or worse) DSM data, typically from satellite imagery.

The Google Maps Platform team is constantly working to improve coverage for our API services. The map below shows where **HIGH** and **MEDIUM** quality solar data is available. It does not show where **LOW** quality solar data is available.

Drag the map to show a specific area and then zoom in for a detailed view.

Note: The coverage data shown here is an approximation of the exact coverage. Download the shape files used to create this map as a ZIP file.

Was this helpful?

Recommended for you

Set up your Google Cloud project

This guide shows how to set up your Google Cloud project before using the Google Maps Platform APIs. While you may have completed some of these steps in the Getting started with Google Maps Platform page, this topic provides additional, useful

Updated Oct 5, 2023

Make a building insights request

The buildingInsights endpoint provides insights about the location, dimensions, and solar potential of a building. In particular, you can get information about: To learn more about how the Solar API defines solar potential and sunniness, see [Solar](#)


Updated Oct 5, 2023

Solar API Usage and Billing

The Solar API uses a pay-as-you-go pricing model. Solar API requests generate calls to one SKU for all but mobile-native apps. Along with the overall Google Terms of Use, there are usage limits specific to the Solar API. Manage your costs and usage

Updated Oct 5, 2023

Solar API Concepts

 developers.google.com/maps/documentation/solar/concepts

- Home
- Products
- Google Maps Platform
- Documentation
- Solar API

Was this helpful?

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

The Solar API provides solar potential data through the `buildingInsights` and `dataLayers` endpoints. To use Solar API data, understanding the following concepts may be helpful:

- Solar irradiance and insolation
- Sunniness and sunshine quantiles
- Rasters
- Flux

Solar irradiance and insolation

A building's solar potential is largely based on the amount of sunlight it receives, along with other factors. *Solar irradiance* is the amount of light that falls on a given area, while *solar insolation* is a measurement of the average solar irradiance an area receives over time.

A *kilowatt* (kW) is a measure of *power*, or the rate at which something uses energy, while a *kilowatt-hour* (kWh) is a measure of *energy* used or energy capacity. Solar irradiance is measured in kilowatts, while solar insolation is measured in kilowatt-hours.

1 kWh/kW equals 1 sun hour, which is defined as one hour where the intensity of sunlight reaches an average of 1,000 Watts (1 kilowatt) of energy per square meter.

For example, if a part of a roof has a solar insolation of 2000 kWh/kW/year, a 1 kW solar panel array placed there will produce 2000 kWh/year. A 4 kW array placed in the same location will produce 8000 kWh/year.

Standard Test Conditions are an industry standard benchmark used to determine solar panel power output. At STC, the amount of power a solar panel outputs becomes its maximum power rating, or capacity. A 1 kW panel will generate 1 kWh of energy under STC.

Sunniness and sunshine quantiles

The Solar API defines "sunniness" as the level of sunlight received by a particular section of a roof relative to the rest of the roof, annually on average. Some parts of a roof may be darker than others, due to shade from nearby buildings or tree cover, while other parts of a roof may be fully exposed to the sky at all times and therefore receive more sunlight.

The `sunshineQuantiles` field in the `buildingInsights` response provides 11 buckets, or deciles, of the sunniness of a roof or part of a roof. The Solar API takes all of the points on the roof, sorts them by their "sunniness," and identifies the highest, lowest, and 9 intermediate evenly spaced values.

For example, assume that the sunniest part (1%) of a given roof receives 1100 kWh/kW/year, while the darkest part (also 1%) of the same roof receives 400 kWh/kW/year. The next darkest 20% of the roof receives 500 kWh/kW/year. The next sunniest 50% of the roof receives 900 kWh/kW/year. The remaining 28% receives 1000 kWh/kW/year.

Rasters

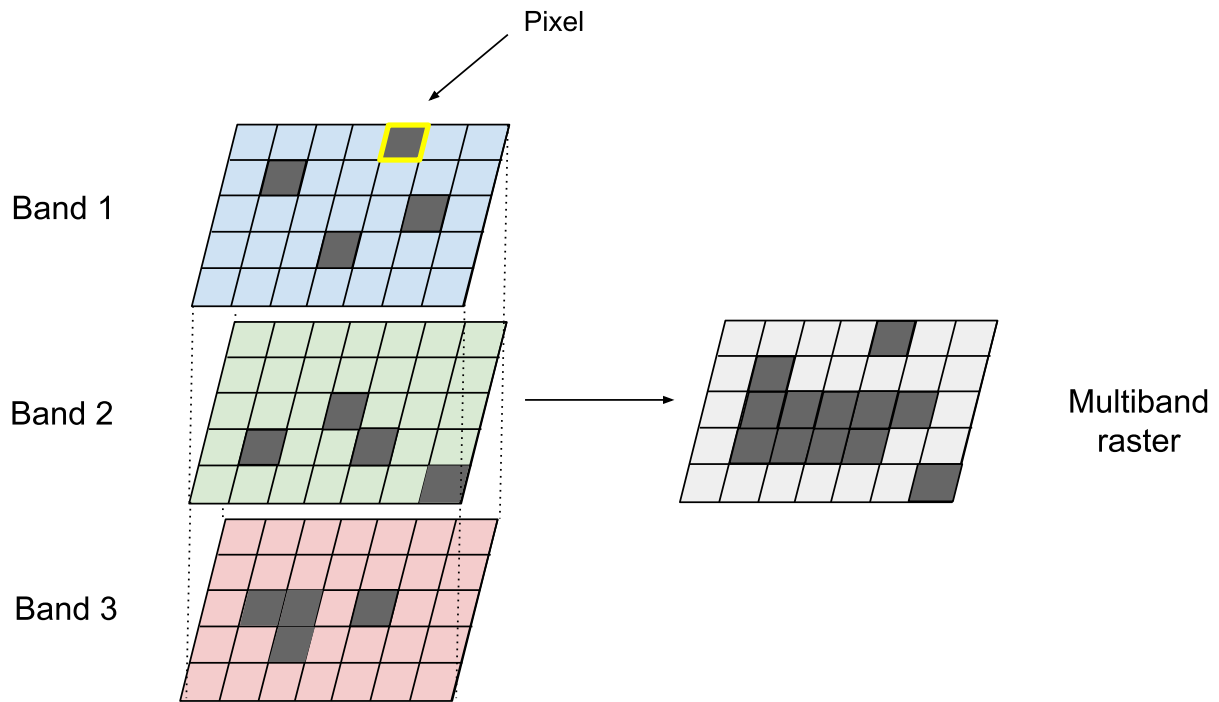
The `dataLayers` endpoint returns solar information encoded in GeoTIFFs, which are a type of raster.

A *raster* is composed of a matrix of cells, or *pixels*, arranged in rows and columns. Each pixel contains a value that represents information about that location, such as elevation, tree canopy, sunlight, among others.

Rasters store *discrete* and *continuous* data. *Discrete* data, such as land cover or soil type, is thematic, or categorical. *Continuous* data represents phenomena that have no clear boundaries, such as elevation or aerial imagery.

Rasters are composed of *bands*, which measure different characteristics of a dataset. Rasters can have a single band or multiple bands. Each band is composed of a matrix of cells, or *pixels*, which store information. Pixels can store float or integer values.

The *bit depth* of a pixel indicates the number of values that a pixel can store, based on the formula 2^n , where n is the bit depth. For example, an 8-bit pixel can store up to 256 (2^8) values ranging from 0 to 255.



Flux

You can request flux maps using the `dataLayers` endpoint. The Solar API defines *flux* as the annual amount of sunlight on roofs in kWh/kW/year. In calculating flux, the Solar API takes the following variables into account:

- **Location information:** The Solar API uses hourly solar irradiance data from various weather sets, which are typically on a 4 to 10 km grid. The API computes the position of the sun in the sky at each hour of the year. This is location-dependent and may vary as a result.
- **Weather patterns (clouds):** These are accounted for in the solar irradiance data.
- **Shade from nearby obstacles:** Shading from trees, other buildings, and other parts of the roof are taken into account in calculations.
- **Orientation:** The pitch and azimuth of each part of the roof.
- **True efficiency:** The values computed by the Solar API are independent of the panel efficiency. To calculate energy production, you must multiply by the kilowattage of the panels and factor in other system losses. For more information, see [Calculate solar costs and savings](#).

The Solar API does not take the following variables into account:

- **Inverter efficiency and other losses:** Most values are computed in DC kWh, but some are converted to AC kWh assuming an 85% system efficiency.
- **Soiling and snow:** These are not included in the calculations.

Was this helpful?

Recommended for you

Set up your Google Cloud project

This guide shows how to set up your Google Cloud project before using the Google Maps Platform APIs. While you may have completed some of these steps in the Getting started with Google Maps Platform page, this topic provides additional, useful

Updated Oct 5, 2023

Make a building insights request

The buildingInsights endpoint provides insights about the location, dimensions, and solar potential of a building. In particular, you can get information about: To learn more about how the Solar API defines solar potential and sunniness, see Solar


Updated Oct 5, 2023

Make a data layers request

The dataLayers endpoint provides detailed solar information for a region surrounding a specified location. The endpoint returns 17 downloadable TIFF files, including: For more information about how the Solar API defines flux, see Solar API Concepts.

Updated Oct 5, 2023

Solar API overview

 developers.google.com/maps/documentation/solar/overview

- Home
- Products
- Google Maps Platform
- Documentation
- Solar API

Was this helpful?

bookmark_border Stay organized with collections Save and categorize content based on your preferences.

The Google Maps Platform Solar API is a service focused on helping accelerate solar and energy system installations. The Solar API makes detailed rooftop data based on Google's expansive mapping and computing resources available to help estimate renewable rooftop solar energy potential and savings.

Why use the Solar API

The API accepts requests for three endpoints:

- **buildingInsights:** This service endpoint returns insights about the location, dimensions, and solar potential of a building.
- **dataLayers:** This service endpoint returns URLs for raw solar information datasets for an area surrounding a location.
- **geoTiff:** This endpoint fetches rasters with encoded solar information, including a digital surface model, an aerial image, annual and monthly flux maps, and hourly shade.

This data can help users:

- Remotely design a solar power system
- Reduce solar site assessment time
- Prioritize installation locations
- Create more accurate proposals
- Increase customer conversion rates
- Provide insightful details to educate consumers

Solar API country and region coverage

See Solar API supported countries and regions for the latest coverage details, on a country-by-country basis.

How to use the Solar API

- | | | |
|---|---|--|
| 1 | Get set up | Start with Set up your Google Cloud project and complete the instructions that follow. |
| 2 | Get insights about the location, dimensions and solar potential of a building. | See Make a Building Insights request. |
| 3 | Get raw solar information for an area surrounding a location. | See Make a Data Layers request. |

What's next

Was this helpful?

Recommended for you

Set up your Google Cloud project

This guide shows how to set up your Google Cloud project before using the Google Maps Platform APIs. While you may have completed some of these steps in the Getting started with Google Maps Platform page, this topic provides additional, useful

Updated Oct 5, 2023

Make a building insights request

The buildingInsights endpoint provides insights about the location, dimensions, and solar potential of a building. In particular, you can get information about: To learn more about how the Solar API defines solar potential and sunniness, see Solar

Updated Oct 5, 2023

Solar API Usage and Billing

The Solar API uses a pay-as-you-go pricing model. Solar API requests generate calls to one SKU for all but mobile-native apps. Along with the overall Google Terms of Use, there are usage limits specific to the Solar API. Manage your costs and usage