Protogrid

JSON API Documentation

Protogrid offers a REST-API in order to use the Protogrid services from external applications.

Before You Start

The Protogrid API is designed for anyone who's comfortable creating custom-coded solutions or integrating with RESTful APIs (usually developers or software engineers). If you think you may need some help integrating with the Protogrid API, ask our Protogrid experts by mail protogrid-support@protogrid.com to help develop a custom tool for you. Please note that only documented API features are supported.

REST is an architectural style for programming interfaces that leads to no intermediate state about each client on the server. In other words, each client needs to take care of its own session data, such as authentication handles.

Please note the pricing information concerning the available data volume.

Structure of an http request

An http request consists of the following components:

Base address: the root URL of your Protogrid environment
Example: https://your-environment-name.protogrid.com
The API endpoint (follows directly after the base address to form the request URL)
Example: /api/v2/apps
A HTTP Header field containing the cookie
Details: see below
Either GET or POST as the http method.

Authentication

At the moment only Basic Authentication is supported. If you need OAuth2 please contact us at protogrid-support@protogrid.com. All http requests require an authentication. If the authentication fails, an error will be returned. The error describes whether a HTTP Header field is missing or the login data is incorrect.

How to authenticate

A POST http request to the API endpoint **"/api/v2/authenticate"** does log in the user and the response contains the set-cookie header with the generated cookie.
HTTP Header fields:
<user_id> [required]: The user_id corresponds to the username or e-mail address of the user.
Either <user_email> or <user_id> have to be specified.
<user_email> [required]: The user_email corresponds to the e-mail address of the user. Either
<user_email> or <user_id> have to be specified.
<user_secret> [required]: The user_secret corresponds to the password of the user.

Example request: https://example.protogrid.com/api/v2/authenticate
Request Header:

```
{
        POST /api/v2/authenticate
        Host: your_environment.protogrid.com
        user_id: example_user
        user_secret: example_secret
}
```

Example in jQuery:

```
$.ajax({
   type:'POST',
   url: 'https://example.protogrid.com/api/v2/authenticate',
   contentType: 'application/json; charset=utf-8',
   dataType: 'json',
   beforeSend: function(xhr){
      xhr.setRequestHeader('user_id','tester@test.com');
      xhr.setRequestHeader('user_secret','test_password');
   }
});
```

Example in Python (with requests):

```
url = "https://example.protogrid.com/api/v2/authenticate"
headers = dict(user_id="test_user@testdomain.com", user_secret="test_password")
req = requests.post(url, headers=headers)
response = req.text
response = json.loads(response)
cookie = req.cookies['session']
```

Example response of successful authentication:

```
{
        "errors": [],
        "protogrid_environment_version": "1.3.9",
        "result": "Login successful!"
}
```

Example response of unsuccessful authentication:

```
{
        "errors": [
                {
```

```
                    “code”: 401,
                    “message”: “Your login wasn’t recognized. Please check your e-mail
                    address and password.”
                }
            ],
            "protogrid_environment_version": "1.3.9",
            “result”: {}
}
```


How to send authenticated http requests

Each request to the API must be authenticated using the cookie the /api/v2/authenticate API endpoint returns.

Example ajax request:
Note that when jQuery runs in a browser, that the cookie is passed automatically with the request.

```
$jq.ajax({
        type: 'GET',
        url: 'https://example.protogrid.com/api/v2/apps',
        contentType: 'application/json; charset=utf-8',
        dataType: 'json',
        success: function(data) {
                console.log(data);
        },
        error: function(data) { console.log(data); }
});
```

Note: Some browsers handle cookies differently. For more information, see the browser specific documentation.

Example Python request:
The cookie variable was set above in the authenticate example.

```
url = "https://example.protogrid.com/api/v2/apps"
req = requests.get(url, cookies=cookie)
response = req.text
response = json.loads(response)
```

Note: For more information about the requests, please refer to http://docs.python-requests.org/en/master/

API Endpoints

This section lists the different API endpoints you can use to access or update Protogrid data. The respective http method is defined as [METHOD] following the endpoint in this documentation.

Every request to such an API endpoint returns a JSON object. The returned JSON object always contains an error and a result object. If the request was successful, the result object contains the requested or updated data. When the request fails, the errors object contains the error messages which occurred when trying to execute the request.

Example successful response:

```
{
        "errors": [],
        "protogrid_environment_version": "1.3.9",
        "result":{
                “key”: {…},
                “key”: […],
                “key”: “data”
        }
}
```

Example failure response:

```
{
        “errors”: [
                { … },
                { … }
        ],
        "protogrid_environment_version": "1.3.9",
        “result”: {}
}
```

The Protogrid JSON API offers the following API endpoints:

/api/v2/apps

[GET] Returns a list of all applications to which the authenticated user has access.
Example request: https://example.protogrid.com/api/v2/apps
Example response:

```
{

        "errors": [],
        "protogrid_environment_version": "1.3.9",
        "result":[
                {
                        "app_id": "app_example_id",
                        "description": "Example App",
                        "logo_url": "",
                        "theme_color": "blue-dark",
```

```
                "title": "Example App ",
                "url": "/example",
                "url_name": "example"
            },
            {
                "app_id": "appID",
                "description": "Your awesome app",
                …
            },
            …
        ]
}
```


/api/v2/apps/<app_name>

[GET] Returns basic information about one application.
Example request: https://example.protogrid.com/api/v2/apps/example
Example response:


```
{
        "errors": [],
        "protogrid_environment_version": "1.3.9",
        "result": {
            "app_id": "appID",
            "description": "Example App",
            "logo_url": "",
            "theme_color": "blue-dark",
            "title": "Example App ",
            "url": "/example",
            "url_name": "example"
        }
}
```


/api/v2/apps/<app_name>/views

[GET] Returns a list of all view names in this application.
Details on how to access a specific view, see here.
Details on what views are supported, see here.
Example request: https://example.protogrid.com/api/v2/apps/example/views
Example response:


```
{
        "errors": [],
        "protogrid_environment_version": "1.3.9",
        "result": {
            "views": [
                {"view_name": "examples_by_id"},
```

```
                {...}
            ]
        }
}
```

/api/v2/apps/<app_name>/views/<view_name>

Most of the API endpoints described above except you to specify App- or Card ids. At the time at which this data is requested, these ids therefore already need to be known. It is therefore necessary to have convenient ways of asking Protogrid about what ids are available based on different search criteria. Database views can be used to acquire collections of data based on sorting and filtering URL parameters.

[GET] Returns all the entries in the specified view. Please note that URL parameters must be URI encoded.
URL Parameter:

- start_key: ["proto_key", "column_key", "filter_value", null]
  where column_key and filter_value are optional.
- end_key: ["proto_key", "column_key", "filter_value", {}]
  where column_key and filter_value are optional
- page_number: Set to 1 if on the first page, otherwise 2
- limit: Accepts a number, which defines how many cards are returned in the result object.
  Example: limit=15
  Default value: 10
- descending: If set to true the results will be in descending order.
  Example: descending=true
  Default value: False

Details: For more details about the URL parameters see the CouchDB documentation.
Full list of views can be found in section available views.
Example request: https://example.protogrid.com/api/v2/apps/example/views/example_view
?start_key=["example_proto_key",null]&end_key=["example_proto_key",{}]&
page_number=1&limit=2&descending=false
The URL parameters need to be URI encoded. So from the example above the start_key parameter would look like start_key= %5B%22example_proto_key%22%2Cnull%5D
Example response:

```
{
    "errors": [],
    "protogrid_environment_version": "1.3.9",
    "result": [
        {"key": "example_document_key", "shortname": "example_shortname"},
        {...}
    ]
}
```

/api/v2/apps/<app_name>/cards

[POST] Creates a new card. It is necessary to specify a new card key in the JSON under the field _id and a proto_key on which the card will be based. If wished, it is possible to send the design_elements along, if the field is undefined, Protogrid uses the design_elements specified in the Proto. Note that obligatory fields need to be set.

Example request: https://example.protogrid.com/api/v2/apps/example/cards

```
JSON: {"_id": "some_card_id",
        "proto_key": "some_proto_key",
        "design_elements":[
            {"definition_key": "example_design_element_key", "value":
            "example_value"},
            {…}
        ]
       }
```

Example response:

```
{
    "errors": [],
    "protogrid_environment_version": "1.3.9",
    "results": {
        "_id: "some_card_id",
        "_rev": "some_rev_id",
        "denormalized": {…},
        …
    }
}
```

/api/v2/apps/<app_name>/cards/<card_key>

[GET] Returns a specific card identified by card_key.
[DELETE] Deletes a specific card identified by card_key logically. The deleted card will be still available, either through the API or the UI using the Trash under Administration. The return value contains the deleted card.
[POST] Updates the card identified by card_key. If the _id field is set in the JSON, it needs to match the card_key. In addition, the proto_key must match those of a proto, otherwise an error is thrown.
Example request: https://example.protogrid.com/api/v2/apps/example/cards/some_card_id

```
JSON: {_id: some_card_id,
        proto_key: some_proto_key,
        design_elements:[
            {"definition_key": "example_design_element_key", "value":
            "example_value"},
            {…}
        ]
       }
```

Example response:

```
{
        "errors": [],
        "protogrid_environment_version": "1.3.9",
        "results": {
                "_id: "some_card_id",
                "_rev": "some_rev_id",
                "denormalized": {…},
                …
        }
}
```


/api/v2/apps/<app_name>/protos

[GET] Returns a list of all protos of the specified application.
The following URL parameter can be used for paging:

- card_key: Passing a card_key will give you a result starting from that key
- limit: Defines how many proto keys are returned. Takes a number between 1 and 1000, default is 10.

Example request:
https://example.protogrid.com/api/v2/apps/example/protos?card_key=ExampleProtoKey&limit=1
Example result:


```
{
        "errors": [],
        "protogrid_environment_version": "1.3.9",
        "result": {
                next_card_key: "NextProtoKey"
                "protos":[
                        {"key": "ExampleProtoKey", "shortname": {"en": "Proto: Example Proto"}}
                ]
        }
}
```


/api/v2/apps/<app_name>/protos/<proto_key>

[GET] Returns detailed information about the specified Proto.
Example request: https://example.protogrid.com/api/v2/apps/example/protos/example_proto
Example result:


```
{
        "errors": [],
        "protogrid_environment_version": "1.3.9",
        "result": {
                "key": <proto_key>,
                "shortname": example_proto,
```

```
            "field_and_widget_keys": [
                {
                    "allow_new_cards": true,
                    "cardinality_setting": 2,
                    "couchdb_viewname": "",
                    "disable_label": false,
                    "display_priority": 7,
                    "element_type": "Relation",
                    "grid_size_setting": 0,
                    "help_text": "",
                    "hidden": false,
                    "label_multilanguage_key": "&&mlkey_design_elements",
                    "multiple_values_setting": 1,
                    "name": "card_design_elements",
                    "parent_application_setting": "",
                    "parent_entity_proto_setting": "DesignElementDefinition",
                    "requirement_setting": 0,
                    "show_card_type": true,
                    "show_created_setting": false,
                    "show_creator_setting": false,
                    "show_modified_setting": false,
                    "show_modifier_setting": false,
                    "show_short_name_setting": false,
                    "show_to_users": 0,
                    "user_enabled_setting": 0,
                    "value": [
                        "design_element_key_1 ",
                        "design_element_key_2",
                        …
                    ],
                    "value_type": "key"
                },
                {…},
                …
            ]
        }
}
```


/api/v2/apps/<app_name>/protos/<proto_key>/card-keys

[GET] Returns the keys for all cards based on the specified proto.
The following URL parameter can be used for paging:

- card_key: Passing a card_key will give you a result starting from that key
- limit: Defines how many card keys are returned. Takes a number between 1 and 1000, default is 10.

Example request:
https://example.protogrid.com/api/v2/apps/example/protos/example_proto/card-keys?card_key=ExampleCardKey&limit=1
Example result:


```
{
        "errors": [],
        "protogrid_environment_version": "1.3.9",
        "result": [
                "card_keys":[
                        "ExampleCardKey"
                ],
                "next_card_key": "NextCardKey"
        ]
}
```


/api/v2/user/user-context

[GET] Returns the user context of the currently logged in user.

Example request:
https://example.protogrid.com/api/v2/user/user-context
Example result:


```
{
        "errors": [],
        "protogrid_environment_version": "1.3.9",
        "result": {
                "datetime_locale": "de",
                "languages": [
                        "de"
                ],
                "navigation_stack": [],
                "preferred_locale": "de-DE",
                "user_card": {
                        "_id": "org.couchdb.user:test_user",
                        "_rev": "test_revision_number",
                        "definition_type": "User",
                        "design_elements": [
                                {
                                    "name": "email",
                                    "value": "tester@test.com"
                                },
                                {
                                    "name": "assigned_roles",
                                    "value": [
                                        "&&applicationCreatorRole"
                                    ]
```

```
                    }
                ],
                "modification_log": [
                    {
                        "time": "2016-09-15T21:34:09",
                        "user": "admin"
                    }
                ],
                "name": "user_key ",
                "roles": [
                    "&&applicationCreatorRole"
                ],
                "type": "user"
            },
            "user_id": "user_key"
        }
}
```


Available Database Listings / Views

For details about listings / views and how to request it, please see section about view endpoints. Please note that all views have the Card id as the last key component due to how Protogrid is implemented on top of CouchDB. In most cases it is useful to specify the start_key and end_key URL parameters with null and {} (empty object) as their last element.

all_by_id

Returns all Cards by id. The row values are null (i.e. unused).

by_id_and_value

All Cards by id. Value is the raw Card.

all_protos_by_id

All Protos by id. Value contains shortname and a list of design_elements

by_id

All non-deleted and non-hidden Cards. Value is null.

by_id_and_value

All non-deleted and non-hidden Cards. Value is raw Card.

deleted_by_id

All deleted Cards by id. Value is null.

by_proto_and_id

All Cards with corresponding Proto by id. Value is null.

by_proto_and_shortname_and_id

All cards with corresponding Proto and shortname by id. Value is null.

by_search_term_and_id

All Cards for a certain serach term by id. Value is null.

deleted_by_search_term_and_id

All deleted Cards for a certain serach term by id. Value is null.

by_proto_and_search_term_and_id

All Protos for a certain search term by id. Value is null.

by_design_element_and_value_and_id

All Cards by design_element, date and id. Value is null.

deleted_by_design_element_and_value_and_id

All deleted Cards by design_element, date and id. Value is null.

by_proto_and_design_element_and_value_and_id

All Cards by underlying Proto, design_element, date and id. Value is null.

relational_definitions_by_id_and_related_proto

All relational definitions by related proto and id. Value is null.

data_protos_by_id

All data Protos by id. Value is null.

data_protos_by_search_term_and_id

All data Protos for a certain search term and by id. Value is null.

datetime_field_definitions_by_id

All datetime field definitions by id.

datetime_field_definitions_by_search_term_and_id

All datetime field_definitions for a certain search term by id. Value is null.

related_keys_by_id

All Cards with related keys by id. Value contains the related keys.

by_relating_key

All Cards with the related key by id. Value is null.

tableview_data_by_id

All Cards contained in a tableview by id. Value contains the raw card.

shortname_by_language_objects_by_id

All Cards which contains shortname data by id. Value contains the shortnames by language.

sums_by_proto_and_design_element

Sums over the values of a field for all Cards of a Proto. Depending on the field type, this uses the following functions:

1. For number fields it produces the sum of values over the field.
2. For text fields it produces the word count over all text entries.
3. For relation fields it produces the number of stored relations.
4. For date/time fields it produces the number of stored date/time values.

sums_by_proto_and_design_element_and_condition

Sums over the values of a field for all Cards of a Proto given a condition from another field. This key requires query keys with four components:

1. The Proto key
2. The design element key for the field to sum over
3. The design element key for the condition field
4. The conditional value in the field specified in component (3)

Example: What is the sum over all values in the “Total” field in the Proto “Invoice”, given that the relation field “customer” holds the key for “John Smith”?

Depending on the field type, this uses the following functions:

1. For number fields it produces the sum of values over the field.
2. For text fields it produces the word count over all text entries.
3. For relation fields it produces the number of stored relations.

4. For date/time fields it produces the number of stored date/time values.

Revisionsgeschichte dieses Dokuments

Datum	Version	Autoren	Kurzbeschreibung
15.04.2016	0.9	RBR	Erste Version
17.04.2016	1.0	MMU, RBR	Review durchgeführt
16.05.2016	1.1	HMA, MMU	2. Review durchgeführt