The Climate Equity Reference Calculator Web API

Climate Equity Reference Calculator Team
[mailto:calculator@climateequityreference.org]

November 2012, updated September 2015, version 1.1

Table of Contents

Introduction

The Climate Equity Reference Calculator (CERc) web API allows web applications to run the CERc calculator and capture the outputs without having to install any code or display the CERc user interface page. To use the API you must be familiar with web programming.

The CERc web API uses **GET** and **POST**. To access the API, point to `http://calculator.climateequityreference.org/api`.

The basic steps are either:

1. **GET** a database to use for a session, set user-specific parameter values using **POST**, and then repeatedly use **POST** to get results, or

2. Use **POST**, with no database, which will create a temporary database on each call.

The first version is much faster for repeated queries than the second one, especially if there is no need to change the parameter values after they first set.

Except for error messages and the response to the HTTP **OPTIONS** command (which simply says that **GET** and **POST** are the only options), all responses are JSON. Errors and the response to the **OPTIONS** command are plain text. If there is no error then the header is sent with the status `200 OK`. Any other status signals an error.

The API behavior changes from time to time. If you want to force the behavior of a certain API version, pass the version number to the API with the `v` parameter, for example **`v=1.1`**.

GET

Use GET to either create a new database and get its location or to query parameter values. Here are the available commands, with sample responses.

Create a new database

`GET` [q=new_db]

```
prompt% curl --get "http://calculator.climateequityreference.org/api/?q=new_db"
{"db":"fw-sql3-DvaS6N"}
```

Get parameter values

The parameter values are set using the **POST** command. If the user changes parameter values, then the new values are stored in the user's copy of the database. This can be done anonymously, in which case the parameter values cannot be recovered. But if a database name is obtained using **`q=new_db`**, then the values are persistent, and can be recovered using **GET**.

Default values

The default parameter values apply unless they are overridden using **POST**.

`GET [q=params]`

```
prompt% curl --get "http://calculator.climateequityreference.org/api/?q=params"
```


```
{"cum_since_yr":{"description":"The year when historical responsibility begins",
"advanced":false,"db_param":"cumsince","value":1990,"min":1850,"max":2010,"step"
:10,"list":null,"type":"int"},"use_lulucf":{"description":"Include land-use emis
sions in baseline (from 1950 only)","advanced":false,"db_param":"use_lulucf","va
lue":0,"min":0,"max":1,"step":1,"list":null,"type":"int"},"use_netexports":{"des
cription":"Include emissions embodied in traded goods","advanced":false,"db_para
m":"use_netexports","value":0,"min":0,"max":1,"step":1,"list":null,"type":"int"}
,"use_nonco2":{"description":"Include non-CO2 gases in baseline (from 1990 only)
...
```


User-specified values

User-specified values are stored in the user's copy of the database.

`GET [q=params] [db=`*`dbname`*`]`

```
prompt% curl --get "http://calculator.climateequityreference.org/api/?q=params&
db=fw-sql3-DvaS6N"
```


```
{"cum_since_yr":{"description":"The year when historical responsibility begins",
"advanced":false,"db_param":"cumsince","value":1900,"min":1850,"max":2010,"step"
:10,"list":null,"type":"int"},"use_lulucf":{"description":"Include land-use emis
sions in baseline (from 1950 only)","advanced":false,"db_param":"use_lulucf","va
lue":0,"min":0,"max":1,"step":1,"list":null,"type":"int"},"use_netexports":{"des
cription":"Include emissions embodied in traded goods","advanced":false,"db_para
m":"use_netexports","value":0,"min":0,"max":1,"step":1,"list":null,"type":"int"}
...
```


Get the valid range of years

Depending on the baseline and emergency pathway, the valid range of years might be longer or shorter. To get the valid range of years for either the default or user-specific settings, use

`GET [q=year_range] [db=`*`dbname`*`]`

```
prompt% curl --get "http://calculator.climateequityreference.org/api/?q=year_range"
```


```
{"min_year":"1990","max_year":"2030"}
```


```
prompt% curl --get "http://calculator.climateequityreference.org/api/?q=year_range&
db=fw-sql3-DvaS6N"
```


```
{"min_year":"1850","max_year":"2050"}
```


Get general information about the calculator

The calculator has general information that cannot be changed by the user. Consequently these should normally be called without specifying a database. However, as indicated for each command, it is possible to specify a database, which can be useful for debugging or to ensure that a default database has not been updated.

List of available pathways

When running the calculator, it is possible to pass one of a number of emergency pathways. The list is provided using **`q=params`**, but only by display name in order of ID number. This call provides an alternative way to get the list that returns

the ID number, short code, and display name. When setting the *emergency_path* parameter, use the ID number.

GET [q=pathways] [db=*dbname*]

```
prompt% curl --get "http://calculator.climateequityreference.org/api/?q=pathways"
```


```
[{"id":"13","advanced":"0","short_code":"1.5Cmarkerpathway","display_name":"1.5
&#8451 pathway"},{"id":"14","advanced":"0","short_code":"2.0Cmarkerpathway","di
splay_name":"2&#8451 pathway"},{"id":"15","advanced":"1","short_code":"G8pathwa
y","display_name":"G8 pathway"}]
```


Version number for the database or calculator

User databases expire if they are not used for 24 hours, but the database or calculator code could still change over that time if a new version is being released. It is good practice to check, at the start of each session, that the default and user database version numbers are the same. This can be done by using **q=data_ver** with and without specifying a database.

GET [q=calc_ver] [db=*dbname*]

GET [q=data_ver] [db=*dbname*]

```
prompt% curl --get "http://calculator.climateequityreference.org/api/?q=calc_ver"
```

```
"3.0.0"
```

```
prompt% curl --get "http://calculator.climateequityreference.org/api/?q=data_ver&
db=fw-sql3-DvaS6N"
```

```
"7.0.0"
```

ISO 3-letter country codes recognized by the calculator

The calculator API reports national emissions labeled by almost-standard ISO 3-letter country codes. One notable difference is China, where the calculator combines data for mainland China (CHN) and Hong Kong (HKG); the combination is labeled CHK. Because the calculator codes are not fully standard, and because the standard itself has changed over time, it can be useful to check the definitions used by the calculator and API. Also, for presentation it is useful to have the country names corresponding to the codes. The **q=countries** query provides the list.

GET [q=regions] [db=*dbname*]

```
prompt% curl --get "http://calculator.climateequityreference.org/api/?q=countries"
```


```
[{"iso3":"ALB","name":"Albania"},{"iso3":"DZA","name":"Algeria"},{"iso3":"AGO","
name":"Angola"},{"iso3":"ATG","name":"Antigua and Barbuda"},{"iso3":"ARG","name"
:"Argentina"},{"iso3":"ARM","name":"Armenia"},{"iso3":"AUS","name":"Australia"},
{"iso3":"AUT","name":"Austria"},{"iso3":"AZE","name":"Azerbaijan"},{"iso3":"BHS"
,"name":"Bahamas, The"},{"iso3":"BHR","name":"Bahrain"},{"iso3":"BGD","name":"Ba
ngladesh"},{"iso3":"BRB","name":"Barbados"},{"iso3":"BLR","name":"Belarus"},{"is
o3":"BEL","name":"Belgium"},{"iso3":"BLZ","name":"Belize"},{"iso3":"BEN","name":
"Benin"},{"iso3":"BTN","name":"Bhutan"},{"iso3":"BOL","name":"Bolivia"},{"iso3":
"BIH","name":"Bosnia and Herzegovina"},{"iso3":"BWA","name":"Botswana"},{"iso3":
...
```


List of region names and codes used by the calculator

The regions defined by the calculator are for the most part commonly-used regions. However, unlike for countries there is no set of ISO standard region codes. The **q=regions** query provides the list of region codes and names that the calculator recognizes. If a country code is passed to this command, then the calculator only returns those regions that contain that country.

GET [q=regions] [country=*iso3*] [db=*dbname*]

```
prompt% curl --get "http://calculator.climateequityreference.org/api/?q=regions"
```


```
[{"region_code":"world","name":"World"},{"region_code":"high_income","name":"High
Income"},{"region_code":"upper_mid_income","name":"Upper Middle Income"},{"region
_code":"lower_mid_income","name":"Lower Middle Income"},{"region_code":"low_incom
e","name":"Low Income"},{"region_code":"annex_1","name":"Annex 1"},{"region_code"
:"annex_2","name":"Annex 2"},{"region_code":"non_annex_1","name":"Non-Annex 1"},{
"region_code":"eit","name":"EITs"},{"region_code":"ldc","name":"LDCs"},{"region_c
```

ode":"eu15","name":"EU 15"},{"region_code":"eu13","name":"EU 13"},{"region_code":
"eu28","name":"EU 28"},{"region_code":"OECD_NA","name":"OECD North America"},{"re
gion_code":"OECD","name":"OECD"},{"region_code":"OECD_Europe","name":"OECD Europe
"},{"region_code":"OECD_Pacific","name":"OECD Pacific"},{"region_code":"EE_Eurasi
...

prompt% curl --get "http://calculator.climateequityreference.org/api/?q=regions&country=USA"

[{"region_code":"high_income","name":"High Income"},{"region_code":"annex_1","nam
e":"Annex 1"},{"region_code":"annex_2","name":"Annex 2"},{"region_code":"OECD_NA"
,"name":"OECD North America"},{"region_code":"OECD","name":"OECD"},{"region_code"
:"NAMER","name":"North America"},{"region_code":"OECD90","name":"OECD 1990 member
s"}]

POST

There are several options for **POST** which allow you to set parameters and get results.

`POST` [reset=`yes`] [db=*`dbname`*] [years=*`yearlist`*] [countries=*`countrylist`*] [*param1*=*`val1`*] [...] [*paramN*=*`valN`*]

- The *`yearlist`* can be a single year, several years separated by commas, a range separated by colon, or a combination. For example, **`years=1990:1995,2020,2030`**.
- The countrylist is a comma-separated list of country ISO 3-letter codes or CERc region codes. (You can get the list of recognized ISO 3-letter codes and region codes using **GET**, as described above.) For example, **`countries=USA,BRA,annex_1,eu28`**.

Here are some examples of using **POST**.

Example 1. Run with defaults and get all values

prompt% curl http://calculator.climateequityreference.org/api/ -d ""

[{"code":"AFG","name":"Afghanistan","year":"1950","pop_mln":"8.149999213","gdp_b
lnUSDMER":"17.57187384","gdp_blnUSDPPP":"37.9472589866867","fossil_CO2_MtCO2":"0
.0843333333333333","LULUCF_MtCO2":"6.32063547133333","NonCO2_MtCO2e":"0.0","vol_
rdxn_MtCO2":null,"allocation_MtCO2":"0.0843333333333333","responsibility_MtCO2":
"0.0120192925576376","r_frac":"6.60533089406693e-06","a1_dom_rdxn_MtCO2":"0.0","
net_import_MtCO2":"0.0","capacity_blnUSDMER":"2.50436552334616","c_frac":"0.0011
1221320982051","rci":"0.000559409270357291","pop_mln_above_dl":"1.26635216481992
","pop_mln_above_lux":"1.64199178219698e-05","lux_emiss_MtCO2":"6.82149867379303
e-07","lux_emiss_applied_MtCO2":"0.0","kyoto_gap_MtCO2":"0.0"},{"code":"AFG","na
me":"Afghanistan","year":"1950","pop_mln":"8.149999213","gdp_blnUSDMER":"17.5718
7384","gdp_blnUSDPPP":"37.9472589866867","fossil_CO2_MtCO2":"0.0843333333333333"
...

Example 2. Run with defaults and get values only for Annex I in 2020

prompt% curl http://calculator.climateequityreference.org/api/ -d "years=2020&countries=annex_1"

[{"code":"annex_1","name":"Annex 1","year":"2020","pop_mln":"1331.970229","gdp_bl
nUSDMER":"50496.842147372","gdp_blnUSDPPP":"44193.4919882075","fossil_CO2_MtCO2":
"14700.5310366383","LULUCF_MtCO2":"-73.7032638298722","NonCO2_MtCO2e":"3503.47667
093233","vol_rdxn_MtCO2":"0.0","allocation_MtCO2":"5081.62979431172","responsibil
ity_MtCO2":"612934.618631315","r_frac":"0.727629079987297","a1_dom_rdxn_MtCO2":"0
.0","net_import_MtCO2":"2134.9619323928","capacity_blnUSDMER":"39969.7763429238",
"c_frac":"0.709397421911075","rci":"0.743301850675701","pop_mln_above_dl":"1203.9
7722628009","pop_mln_above_lux":"324.777148950213","lux_emiss_MtCO2":"3988.866014
99692","lux_emiss_applied_MtCO2":"0.0","kyoto_gap_MtCO2":"1176.94831227618"}]

Example 3. Run with a specified database, change a parameter, and get values for Annex I in 2020

```
prompt% curl http://calculator.climateequityreference.org/api/ -d "db=fw-sql3-
DvaS6N&cum_since_yr=1900&years=2020&countries=annex_1"
```


```
[{"code":"annex_1","name":"Annex 1","year":"2020","pop_mln":"1331.970229","gdp_bl
nUSDMER":"50496.842147372","gdp_blnUSDPPP":"44193.4919882075","fossil_CO2_MtCO2":
"14700.5310366383","LULUCF_MtCO2":"-73.7032638298722","NonCO2_MtCO2e":"3503.47667
093233","vol_rdxn_MtCO2":"0.0","allocation_MtCO2":"4977.44608918771","responsibil
ity_MtCO2":"655095.994747327","r_frac":"0.739930839663735","a1_dom_rdxn_MtCO2":"0
.0","net_import_MtCO2":"2134.9619323928","capacity_blnUSDMER":"39969.7763429238",
"c_frac":"0.709397421911075","rci":"0.749203215606724","pop_mln_above_dl":"1203.9
7722628009","pop_mln_above_lux":"324.777148950213","lux_emiss_MtCO2":"3988.866014
99692","lux_emiss_applied_MtCO2":"0.0","kyoto_gap_MtCO2":"1176.94831227618"}]
```


Example 4. Run with a specified database and emergency pathway, and get values for Annex I in 2020

```
prompt% curl http://calculator.climateequityreference.org/api/ -d "db=fw-sql3-
TbGUvI&emergency_path=13&years=2020&countries=annex_1"
```


```
[{"code":"annex_1","name":"Annex 1","year":"2020","pop_mln":"1331.970229","gdp_bl
nUSDMER":"50496.842147372","gdp_blnUSDPPP":"44193.4919882075","fossil_CO2_MtCO2":
"14700.5310366383","LULUCF_MtCO2":"-73.7032638298722","NonCO2_MtCO2e":"3503.47667
093233","vol_rdxn_MtCO2":"0.0","allocation_MtCO2":"5081.62979431172","responsibil
ity_MtCO2":"612934.618631315","r_frac":"0.727629079987297","a1_dom_rdxn_MtCO2":"0
.0","net_import_MtCO2":"2134.9619323928","capacity_blnUSDMER":"39969.7763429238",
"c_frac":"0.709397421911075","rci":"0.743301850675701","pop_mln_above_dl":"1203.9
7722628009","pop_mln_above_lux":"324.777148950213","lux_emiss_MtCO2":"3988.866014
99692","lux_emiss_applied_MtCO2":"0.0","kyoto_gap_MtCO2":"1176.94831227618"}]
```


Error Messages

Only two error codes can be returned.

- 410 Gone: If the requested database named in a **db=dbname** parameter does not exist, a 410 code is returned, along with text saying the database no longer exists.
- 400 Bad request: (**GET** only): If the parameters passed to the API are invalid, a 400 code is returned, with text saying what the valid parameters are.

An Example Using PHP

The API can be called from any language that supports HTTP client functions, which means it can be used from almost any modern programming language. Also, there are several libraries and compiled extensions for PHP for making HTTP transactions. For concreteness this example uses a specific PHP library. the PEAR HTTP_Request library. To keep the example as simple as possible the script does not feature some good coding practices that would make it more complex (although shorter). Specifically, there is redundant code and code that could be encapsulated in a class.

To see the example in action, go to http://calculator.climateequityreference.org/api/example.php.

```
<?php
require_once "HTTP/Request.php";

function get_countries() {
    $URL = "http://calculator.climateequityreference.org/api/");
    $req =& new HTTP_Request($URL . "?q=countries");
    $req->setMethod(HTTP_REQUEST_METHOD_GET);
    if (!PEAR::isError($req->sendRequest())) {
        // Note: json_decode returns arrays as StdClass, so have to cast
        $response = (array) json_decode($req->getResponseBody());
    } else {
        throw new Exception($req->getMessage());
    }
    return $response;
}
```

```
function get_params() {
    $URL = "http://calculator.climateequityreference.org/api/");
    $req =& new HTTP_Request($URL . "?q=params");
    $req->setMethod(HTTP_REQUEST_METHOD_GET);
    if (!PEAR::isError($req->sendRequest())) {
        $response = (array) json_decode($req->getResponseBody());
    } else {
        throw new Exception($req->getMessage());
    }
    return $response;
}

function get_country_data($iso3) {
    $URL = "http://calculator.climateequityreference.org/api/");
    $req =& new HTTP_Request($URL);
    $req->setMethod(HTTP_REQUEST_METHOD_POST);
    $req->addPostData('years', 2020); // Note hard-coded year
    $req->addPostData('countries', $iso3);
    if (!PEAR::isError($req->sendRequest())) {
        $response = json_decode($req->getResponseBody());
        // Oddly, the decode procedure duplicates the first element.
        // Test by comparing to the number of elements we expect (1).
        if (count($response) > 1) {
           $response = array_slice($response, 1);
        }
    } else {
        throw new Exception($req->getMessage());
    }
    return $response;
}
?>

<html>
    <head>
        <title>Testing CERc API</title>
    </head>
    <body>
        <h1>Testing CERc API</h1>
        <form method="post">
            <select name="country">
                <?php
                    $countries = get_countries();
                    foreach ($countries as $country) {
                        $ca = (array) $country;
                        $option = '<option value="';
                        $option .= $ca['iso3'];
                        $option .= '">';
                        $option .= $ca['name'];
                        $option .= '</option>';
                        echo $option;
                    }
                ?>
            </select>
            <input type="submit" value="calculate" />
        </form>
        <?php
            if (isset($_POST['country'])) {
                echo '<h2>output</h2>';
                echo '<table>';
                echo '<tr><td>item</td><td>value</td></tr>';
                $country_data_list = get_country_data($_POST['country']);
                $country_data = (array) $country_data_list[0];
                foreach ($country_data as $item=>$value) {
                    $row = '<tr>';
```

```
                    $row .= '<td>' . $item . '</td>';
                    $row .= '<td>' . $value . '</td>';
                    $row .= '</tr>';
                    echo $row;
                }
                echo '</table>';

                echo '<h2>parameters</h2>';
                echo '<table>';
                echo '<tr><td>item</td><td>value</td></tr>';
                $params = get_params();
                foreach ($params as $item=>$info) {
                    $info_array = (array) $info;
                    $row = '<tr>';
                    $row .= '<td>' . $item . '</td>';
                    $row .= '<td>' . $info_array['value'] . '</td>';
                    $row .= '</tr>';
                    echo $row;
                }
                echo '</table>';
            }
        ?>
    </body>
</html>
```