Global Caché IR Database

Full API Specification
Version 1.0

To use the Global Caché IR Database API, a user must be logged in using the **api/accounts/login** call (see below), and the resulting API Key must be passed to the request as a query parameter. While the API Key only needs to be provided when requesting IR Codes, or managing one's account, the API Key query parameter may also be provided on any other calls to the IR Database API as well.

The current URL for Global Caché's IR Database in the cloud is

https://irdb.globalcache.com:8081/

API CALLS (BY URL)

api/brands/

(GET)
Returns a list of available product brands

RETURNED
List of Brands

api/types

(GET)
Returns a list of all available device types

RETURNED
List of Types

api/brands/{brand}/types

(GET)
Returns a list of available device types for the given brand

RETURNED
List of Brands/Types

api/types/{type}/brands

(GET)
Returns a list of available brands for a given device type

Note: This call is not optimized; it may not perform as well as a call to "api/brands/{brand}/types".

RETURNED
List of Brand/Types

api/brands/{brand}/types/{type}/models

(GET)
api/types/{type}/brands/{brand}/models

(GET)
Get models available for the given brand and device type.

RETURNED
List of Models

api/codesets/{setid}?output=email&format=gc&apikey={apikey}

(GET)
Get the full codeset with the given ID.

Note: Must be logged in, and uses one of a limited number of uses per account per day.

For testing purposes, "sandbox=true" can be added, which will return a default dummy codeset instead, but will not consume any of an account's daily allowance.

Acceptable "output" values:

- email – Codes will be sent to the registered e-mail address
- direct – Codes are returned with the response

Acceptable "format" values (Optional - If omitted, both gc and hex will be provided):

- gc – Global Caché sendir format
- compressed – Compressed Global Caché sendir format
- hex – ProntoEdit HEX format

RETURNED
CodeResponse, or List of Codes

api/codesets/{setid}/models

(GET)
Get the model information associated with the codeset with the given ID.

RETURNED
Model

api/codesets/{setid}/functions

(GET)
Get a list of available functions for the codeset with the given ID.

RETURNED
List of Functions

api/codesets/{setid}/functions/{function}/codes?output=email&format=gc&apikey={apikey}

(GET)
Get the code for the given function in the codeset with the given ID.

> Note: Must be logged in, and uses one of a limited number of uses per account per day.
>
> For testing purposes, “sandbox=true” can be added, which will return a default dummy code instead, but will not consume any of an account’s daily allowance.

Acceptable “output” values:

- email – Codes will be sent to the registered e-mail address
- direct – Codes are returned with the response

Acceptable “format” values (Optional - If omitted, both gc and hex will be provided):

- gc – Global Caché sendir format
- compressed – Compressed Global Caché sendir format
- hex – ProntoEdit HEX format

RETURNED
CodeResponse

api/account?apikey={apikey}

Account -> Rel: “self”
(GET)
Returns the current status of the Account with the given key.

RETURNED
Account

api/account/login

(POST)
Content-Type: application/json
Login to an existing account. The returned apikey must be used in subsequent requests to identify the logged in account through a query parameter.

Post Body:
- Email – email address identifying the account.
- Password – The password for the given account

Example:

```
{ "Email":"example@domain.com", "Password":"mypassword" }
```

RETURNED
AccountResponse

api/account/logout?apikey={apikey}

Account -> Rel: “logout”
(POST)
Content-Type: application/json
Logout of the currently logged in account.

RETURNED
AccountResponse

api/account/password?apikey={apikey}

Account -> Rel: “resetpassword”

Content-Type: application/json
(POST)
Change the password of the currently login in account.

Post Body:

- Password – The password for the given account
- NewPassword – The new password to be set for the account.

Example:

```
{"Password":"oldP4ssword","NewPassword":"newP4ssword"}
```

RETURNED
AccountResponse

api/account/recovery

Account -> Rel: “passwordrecovery”
(POST)
Content-Type: application/json
Password recovery. Sends an e-mail to the address passed in with a recovery key to be used to set a new password.

Post Body:

- EmailAddress – The email address of the account.
- Email – An Email Model (see Data Model documentation), specifying the format and text of the e-mail to be sent.
    - The “Body” of the Email must contain the token {{key}}, which will be replaced with the actual account recovery key. An typical option is to embed the key in a URL query, as seen in the example below.

Example:

```
{"EmailAddress":"user@domain.com","Email":{"Subject":"Password
Recovery","Body":"Follow this link to reset your password:
http://www.domain.com/recovery?key={{key}}","FromAddress":"donotreply@domain.com"}}
```

Returned:

AccountResponse (No account information is returned with the response in this case)

api/account/recovery/validate/{key}

(GET)

Validate that a recovery key is valid and current. Recovery Keys are considered invalid after 1 day.

Returned:

AccountResponse (No account information is returned with the response in this case)

api/account/recovery/{key}

(POST)
Content-Type: application/json

Complete the password recovery process. Sets a new password for the user, with a given Recovery Key (from an email sent to the registered e-mail address).

Post Body:

- NewPassword – The new password to be set for the account.

Example:

```
{"NewPassword":"oldP4ssword"}
```

Returned:

AccountResponse

Special character escape sequences:

The following characters require special handling in URLs, particularly in Brand names, Device Types, Model names, and Function names. The following character should be replaced with the sequences as follows:

&	xampx
/	xfslx
>	xgtx
<	xltx
:	xcolx
?	xquex
%	xmodx
+	xaddx

DATA MODELS

AccountResponse

The response to Account POST requests, indicating the status of the requested operation, and provides Account information.

Attributes:

- Status: “success” or “failure”. Whether the request succeeded.
- Message: Message, generally indicates the cause of failures.
- Account: Reflects the current state of the account.

Examples:

```
{"Status":"success","Message":"","Account":{"ApiKey":"NbNpfKChMEuwDHjYRt/egw","Email"
:"tester@globalcache.com","Password":null,"NewPassword":null}}
```

```
{"Status":"failed","Message":"password did not match","Account":null}
```

Account

Used to return relevant account information (especially, an ApiKey), and to send Account information to the API in POST requests.

Attributes:

- ApiKey: the key used to make requests with the logged in account. Api keys are not durable, and will change for each time logged in.
- Email: email address used to identify the account
- Password: Used to login. Always null in responses from the server.
- NewPassword: Used to change the password. Always null in responses from the server.

Links:

- Rel: “register” – Register for a new account, using an email address and password
  - POST – Email and Password required
- Rel: “login” – Login to account, using an email address and password
  - POST – Email and Password required
- Rel: “logout” – Logout of account (apikey query parameter must be present)
  - POST – no post body required
- Rel: “resetpassword” – Change the password (apikey query parameter must be present)
  - POST – Password and NewPassword required

Example:

```
{"ApiKey":"NbNpfKChMEuwDHjYRt/egw","Email":"tester@globalcache.com","Password":null,"
NewPassword":null}
```

CodeResponse

Response message to a call to get a code or codeset, indicating the status of the request.

Attributes:

- Status: “success” or “failure”. Whether the request succeeded.
- Message: Message, generally indicates the cause of failures.
- Code: Code indicating cause of failure.
  - 0 – Success
  - 1 – Unused

- 2 – API Key not found
- 3 – User found, but is not currently logged in
- 4 – Too many IR codes already requested today
- 5 – Unknown output type requested (“email” and “direct” are available)
- 6 – Direct output type is not allowed for this account
- 7 – An apikey is required by the request, and was not provided.
- 8 – Failed to send email, usually due to an invalid email address.
- 9 – Unknown format requested (“hex”, “gc” and “compressed” are allowed, or it may be omitted).

Examples:

```
{"$id":"1","Status":"success","Message":"","Code":0}
```

```
{"Status":"failure","Message":"Maximum number of codes have already been
sent.","Code":4}
```

Code

Represents a single code identified by a SetID and Function. Sent in response to a code request with “output=direct”

Attributes:

- SetID: ID identifying the code set
- Function: Function name identifying the Code within the code set.
- Code1: IR Code, in the requested format, or Global Caché IR format if not specified.
- HexCode1: IR Code, in Hexadecimal IR format, only provided when a format is not specified in the request.
- Code2: Alternate IR code, if applicable.
- HexCode2: Alternate IR code in hex format, if applicable, only provided when a format is not specified in the request.

Examples:

```
{"SetID":"123","Function":"BLAH","Code1":"111111","HexCode1":
"AAAAA","Code2":null,"HexCode2":null}
```

Brand

Represents a Brand, by name

Attributes:
- Name: The brandname

```
{"Name":"Sony”}
```

Links:
- Rel: “types” – List of types for the Brand

Type

Represents a Device type, by name

Attributes:
- Name: The device type’s name

```
{"Name":"TV”}
```

Links:
- Rel: “brands” – List of brands for the Type

Brand/Type

Represents a combination of a Brand and Device Type

Attributes:
- Brand: The brandname
- Type: The device type name

```
{"Brand":"Sony”,"Type":"DVD”}
```

Links:
- Rel: “models” – List of Models for the brand and type.

Model

Represents a specific device, uniquely tied to a particular codeset

Attributes:

- ID: The id of the related codeset
- Name: The Model's name
- Brand: The brand name
- Type: The device type's name
- Notes

```
{"Name":"RDRVDX Series DVDR/VCR Recorder”,"Brand":"Sony”, "Type":"DVD”,"Notes":"Free
text notes go here.”}
```

Links:

- Rel: "functions" – Listing of Functions available for the related codeset
- Rel: "codeset" – Request Code Set to be sent (apikey and output query parameters must be present)

Function

Represents a single function available in a code set.

Attributes:

- Function: The Function's name
- SetID: The ID for the set of codes the function belongs to.

```
{"SetID":"1446","Function":"MEGA"}
```

Links:

- Rel: "functions" – Listing of Functions available for the related codeset
- Rel: "ircode" – Request single IR Code to be sent (apikey and output query parameters must be present)

Email

Model describing an e-mail message's contents.

Attributes:

- Subject: The subject line
- Body: The body of the email. Html is supported in an e-mail body. Replacement keys are enclosed in double curly braces "{{}}", such as used in password recovery keys "{{key}}", and will be replaced with the appropriate information before the e-mail is sent to the user.
- FromAddress: The apparent From address (reply-to) of the sent e-mail.

```
{"EmailAddress":"user@domain.com","Email":{"Subject":"Password
Recovery","Body":"Follow this link to reset your password:
http://www.domain.com/recovery?key={{key}}","FromAddress":"donotreply@domain.com"}}
```

EXAMPLE WORKFLOW:

A user is logged in:

POST /api/account/login

Body –

```
{
      "Email":"joe@company.com",
      "Password":"joesP4ssword"
}
```

Returns –

```
{
      "Status":"success",
      "Message":"",
      "Account":
      {
            "Email":"joe@company.com",
            "ApiKey":" A5zdOEUl0ESLS",
            "Password":null,
            "NewPassword":null
      }
}
```

The API Key must now be provided to any request to send IR Codes, or to change Joe's password, or logout. It is provided as a query parameter: "?apikey=A5zdOEUl0ESLS". It should also be passed to any other queries made to the API, though that is not required.

Next, the full list of brand names is acquired:

GET /api/brands?apikey=A5zdOEUl0ESLS

Returns –

```
[
       {"Name":"Apex"},
       {"Name":"Panasonic"},
       {"Name":"Sony"}
       ...
]
```

Then, to get the list of device types available from Sony:

GET /api/brands/Sony/types?apikey=A5zdOEUl0ESLS

Returns –

```
[
       {"Brand":"Sony", "Type":"DVD"},
       {"Brand":"Sony", "Type":"Laser Disc"},
       {"Brand":"Sony", "Type":"TV"}
       ...
]
```

And, upon choosing to look into DVD models:

GET /api/brands/Sony/types/DVD/models?apikey=A5zdOEUl0ESLS

Returns –

```
[
        {
                "ID":"665",
                "Brand":"Sony",
                "Type":"DVD",
                "Name":"RDRHX Series All Models",
                "Notes":"",
        },{
                "ID":"1103",
                "Brand":"Sony",
                "Type":"DVD",
                "Name":"RDRVDX Series DVDR/VCR Recorder",
                "Notes":""
        }

      ...

]
```

The RDRHX Series Model is selected. The "ID" attribute of the model is used to identify the codeset. Now, either the entire codeset is requested to be emailed to the user now. The output query parameter is provided to specify a type of output:

GET /api/codesets/665?apikey=A5zdOEUI0ESLS&output=email&format=gc

Returns –

```
{
        "Status":"success",
        "Message":"",
        "Code":0,
}
```

Or, a list of available functions is provided (without codes):

GET /api/codesets/665/functions?apikey=A5zdOEUI0ESLS

Returns –

```
[
        {"SetID":"665", "Function":"AUDIO"},
        {"SetID":"665", "Function":"CURSOR UP"},
        {"SetID":"665", "Function":"DIGIT 4"},

      ...

]
```

And then a particular function is chosen to get the IR Code of:

GET
/api/codesets/665/functions/CURSOR%20UP/codes?apikey=A5zdOEUl0ESLS&output=email&format=gc

Returns –

```
{
        "Status":"success",
        "Message":"",
        "Code":0,
}
```

Having now sent the required codes, the user can now be logged out:

POST /api/account/logout?apikey=A5zdOEUl0ESLS

Returns –

```
{

      "Status":"success",

      "Message":"",

      "Account":null

}
```