

MyNetDiary Food Search API v2.7

[The API](#)

[Partner Sign In](#)

[Request](#)

[Response](#)

[Find foods](#)

[Request](#)

[Response](#)

[Get Food Details](#)

[Request](#)

[Response](#)

[UPC \(barcode\) lookup](#)

[Request](#)

[Response](#)

[Data Dictionary](#)

[1. Food](#)

[2. FoodWeight](#)

[3. FoodNutrientValue](#)

[4. Nutrient set](#)

[6. Data Types](#)

[Java Client Example](#)

[Connecting over SSL](#)

[Using Certificates with Java](#)

The API

MyNetDiary API is a supplementary service for licensing partners. You can find all database licensing information including price, samples and technical specs at

<https://www.mynetdiary.com/db>

All foods are available online, you can test data quality and API behavior at [Instant Food Search](#) page.

Partner Sign In

Establish authenticated session before searching foods and getting food details.

Request

HTTPS POST required

URL: **apiPartnerSignin.do**

Parameters:

partnerName - the name identifying MyNetDiary partner company
password - partner's password.

Response

Success:

HTTP Status: 200

JSON: {"message": "Signin success"}

HTTP header named Set-cookie, header value is formatted as "JSESSIONID=..."

Failure:

HTTP Status: 401 or code other than 200

Optional JSON with message explaining the problem

Find foods

Request

URL: **apiFindFoods.do**

HTTP header named Cookie, header value is formatted as "JSESSIONID=..."

Parameters values should be URL encoded:

beanInputString - food search criteria as entered by the user

pageSize - optional maximum number of foods returned in a single response

pageNumber - optional number of page with found foods, starting with 1

highlightedTermClassName - optional parameter,

HTML class name for spans marking found keywords in returned food descriptions.

macrosInSampleServingExpected - optional parameter,

macrosInSampleServingExpected=true results in returning protein, carbs and fat values for the sampleServing, see example below.

Response

Please see [Data Dictionary](#) for field descriptions

HTTP Status 200

JSON example for beanInputString=milk:

```
[
{
  "beanId": 29637105, //foodId
  "descForUi": "<span>Milk</span>", //foodDesc with found words marked with span tag
```

```

    "imageId": 101,
    "sampleServing": { //an example FoodWeight and its calories
      "amount": 1,
      "msreDesc": "cup",
      "gmWgt": 244,
      "calories": 148.83999573
    }
  },
  {
    "descForUi": "2% <span>Milk<Vspan>",
    "imageId": 101,
    "beanId": 29570335,
    "sampleServing": {
      "gmWgt": 244,
      "amount": 1,
      "calories": 121.9999965108,
      "msreDesc": "cup"
    }
  },
  {
    "descForUi": "Chocolate <span>milk<Vspan>",
    "imageId": 122,
    "beanId": 29528328,
    "sampleServing": {
      "gmWgt": 250,
      "amount": 1,
      "calories": 190.000002825,
      "msreDesc": "cup"
    }
  },
  {
    "descForUi": "Tea black regular prepared without <span>milk<Vspan> or sugar",
    "imageId": 26,
    "beanId": 14355,
    "sampleServing": {
      "gmWgt": 29.6,
      "amount": 1,
      "calories": 0.29600000381469727,
      "msreDesc": "fl oz"
    }
  },
  {
    "descForUi": "Silk almond <span class='mycls'>milk<Vspan>",
    "imageId": 101,
    "beanId": 29724830,
    "isGramless": true, //notice that gramless food has no grams in sampleServing
    "sampleServing": {
      "amount": 240,

```

```

    "calories": 60,
    "msreDesc": "ml"
  }
}
]

```

If you need values of protein, carbs and fat you should pass additional parameter macrosInSampleServingExpected=true to apiFindFoods request.

```

{
  "descForUi": "2% <span>Milk</span>",
  "imageId": 101,
  "beanId": 29570335,
  "sampleServing": {
    "gmWgt": 244,
    "amount": 1,
    "carbs": 11.709999663600001,
    "protein": 8.0499997612,
    "fat": 4.8299998616,
    "calories": 121.9999965108,
    "msreDesc": "cup"
  }
},

```

Get Food Details

Request

URL: **apiFoodDetails.do**

HTTP header named Cookie, header value is formatted as "JSESSIONID=..."

Parameter:

foodId - one of beanId values returned in findFoods response

Response

Please see [Data Dictionary](#) for field descriptions

HTTP Status 200

JSON example for foodId=1077:

```

{
  "foodId": 1077,
  "foodDesc": "Milk whole 3.25% milkfat",
  "imageId": 101,

```

```

"retired": false,
"weights": [//complete list of servings
  {"gmWgt": "244.", "amount": 1, "sortOrder": 1, "msreDesc": "cup"},
  {"gmWgt": "15.", "amount": 1, "sortOrder": 2, "msreDesc": "tbsp"},
  {"gmWgt": "31.", "amount": 1, "sortOrder": 3, "msreDesc": "fl oz"},
  {"gmWgt": "976.", "amount": 1, "sortOrder": 4, "msreDesc": "quart"},
  {"gmWgt": "28.35", "amount": 1, "sortOrder": 987, "msreDesc": "oz"},
  {"gmWgt": "1.", "amount": 1, "sortOrder": 988, "msreDesc": "gram"},
  {"gmWgt": "453.592", "amount": 1, "sortOrder": 990, "msreDesc": "lb"}
],
"sampleServing": {"gmWgt": 244, "amount": 1, "calories": 148.84, "msreDesc": "cup"},
"nutrients": [
  // IF food has grams, nutrient values are provided per 100g;
  // ELSE food is gramless: nutrient values are provided per single serving defined for the food.
  {"val": 61, "nutr": 208},
  {"val": 3.25, "nutr": 204},
  {"val": 1.865, "nutr": 606},
  {"val": 0.195, "nutr": 646},
  {"val": 0.812, "nutr": 645},
  {"val": 0, "nutr": 701},
  {"val": 10, "nutr": 601},
  {"val": 43, "nutr": 307},
  {"val": 4.8, "nutr": 205},
  {"val": 0, "nutr": 291},
  {"val": 5, "nutr": 269},
  {"val": 3.15, "nutr": 203},
  {"val": 151.944, "nutr": 318},
  {"val": 0, "nutr": 401},
  {"val": 113, "nutr": 301},
  {"val": 0.03, "nutr": 303},
  {"val": 0.036, "nutr": 415},
  {"val": 0.45, "nutr": 418},
  {"val": 51, "nutr": 324},
  {"val": 0.07, "nutr": 323},
  {"val": 0.3, "nutr": 430},
  {"val": 10, "nutr": 304},
  {"val": 84, "nutr": 305},
  {"val": 132, "nutr": 306},
  {"val": 0.37, "nutr": 309},
  {"val": 0.025, "nutr": 312},
  {"val": 0.004, "nutr": 315},
  {"val": 3.7, "nutr": 317},
  {"val": 0.046, "nutr": 404},

```

```

{"val": 0.169, "nutr": 405},
{"val": 0.089, "nutr": 406},
{"val": 0.373, "nutr": 410},
{"val": 5, "nutr": 417},
{"val": 0, "nutr": 262},
{"val": 0, "nutr": 221},
{"val": 0, "nutr": 295},
{"val": 14.3, "nutr": 421},
{"val": 34.3, "nutr": 314}
]
}

```

UPC (barcode) lookup

Tip: you can quickly test MyNetDiary UPC lookup by using our [free mobile apps](#).

Request

URL: **apiUpcFindFood.do**

HTTP header named Cookie, header value is formatted as "JSESSIONID=..."

Parameter:

upc - food upc code

Response

Please see [Data Dictionary](#) for field descriptions

HTTP Status 200

JSON example for upc=787359177026:

```

{
  "beanId": 19043889,
  "beanDesc": "Honey roasted pecan pieces by fresh gourmet",
  "weights": [
    {"amountId": "1", "itemDesc": "tbsp = 40cals in 7g", "amountMsreDesc": "tbsp", "msreDesc": "tbsp"},
    {"amountId": "987", "itemDesc": "oz = 162cals in 28g", "amountMsreDesc": "oz", "msreDesc": "oz"},
    {"amountId": "988", "itemDesc": "gram = 6cals in 1g", "amountMsreDesc": "gram", "msreDesc": "gram"},
    {"amountId": "990", "itemDesc": "lb = 2592cals in 454g", "amountMsreDesc": "lb", "msreDesc": "lb"},
    {"amountId": "1011", "itemDesc": "ml = 3cals in 0.5g", "amountMsreDesc": "ml", "msreDesc": "ml"}
  ]
}

```

```

    {"amountId": "1012", "itemDesc": "fl oz = 80cals in 14g", "amountMsreDesc": "fl oz",
"msreDesc": "fl oz"},
    {"amountId": "1013", "itemDesc": "teaspoon = 13cals in 2.3g", "amountMsreDesc": "teaspoon",
"msreDesc": "teaspoon"},
    {"amountId": "1015", "itemDesc": "cup = 640cals in 112g", "amountMsreDesc": "cup",
"msreDesc": "cup"}
  ],
  "foodLabel": {
    "FoodName": "Honey roasted pecan pieces by fresh gourmet",
    "Serving": "tbsp (7g)",
    "i": 411, //imageld
    "Calories": "40",
    "FatPercent": "5%",
    "Chol": "0mg",
    "TransFat": "",
    "Calcium": "0%",
    "Fiber": "",
    "TotalCarbsPercent": "1%",
    "Iron": "0%",
    "UnsatFatPoly": "",
    "TotalCarbs": "3g",
    "FiberPercent": "",
    "SugarAlcohol": "0g",
    "Sugars": "2g",
    "CholPercent": "0%",
    "VitaminA": "",
    "CaloriesFromFat": "32",
    "TotalFat": "3.5g",
    "SaturatedFat": "",
    "custom": false,
    "contrib": false,
    "Protein": "0g",
    "Sodium": "25mg",
    "UnsatFatMono": "",
    "SaturatedFatPercent": "",
    "VitaminC": ""
  },
  "nutrients": [
    // IF food has grams, nutrient values are provided per 100g;
    // ELSE food is gramless: nutrient values are provided per single serving defined for the food.
    {"val": 61, "nutr": 208},
    {"val": 3.25, "nutr": 204},
    {"val": 1.865, "nutr": 606},

```

```
{
  "val": 0.195, "nutr": 646},
  "val": 0.812, "nutr": 645},
  "val": 0, "nutr": 701},
  "val": 10, "nutr": 601},
  "val": 43, "nutr": 307},
  "val": 4.8, "nutr": 205},
  "val": 0, "nutr": 291},
  "val": 5, "nutr": 269},
  "val": 3.15, "nutr": 203},
  "val": 151.944, "nutr": 318},
  "val": 0, "nutr": 401},
  "val": 113, "nutr": 301},
  "val": 0.03, "nutr": 303},
  "val": 0.036, "nutr": 415},
  "val": 0.45, "nutr": 418},
  "val": 51, "nutr": 324},
  "val": 0.07, "nutr": 323},
  "val": 0.3, "nutr": 430},
  "val": 10, "nutr": 304},
  "val": 84, "nutr": 305},
  "val": 132, "nutr": 306},
  "val": 0.37, "nutr": 309},
  "val": 0.025, "nutr": 312},
  "val": 0.004, "nutr": 315},
  "val": 3.7, "nutr": 317},
  "val": 0.046, "nutr": 404},
  "val": 0.169, "nutr": 405},
  "val": 0.089, "nutr": 406},
  "val": 0.373, "nutr": 410},
  "val": 5, "nutr": 417},
  "val": 0, "nutr": 262},
  "val": 0, "nutr": 221},
  "val": 0, "nutr": 295},
  "val": 14.3, "nutr": 421},
  "val": 34.3, "nutr": 314}
}
```

Data Dictionary

1. Food

Food header information

Field name	Type	Description
foodId	int	primary key, food identifier
foodDesc	varchar(255)	Food description
imageId	int	An identifier of one of several hundred food icons used by MyNetDiary to create food lists. The data extract includes an archive with food icons. Every icon's filename contains imageId.
retired	boolean	Retired foods are never returned by apiFindFoods, but you can call apiFoodDetails for them.

2. FoodWeight

Food serving information

Field name	Type	Description
foodId	int	primary key, refers to Food.foodId
sortOrder	smallint	primary key, serving identifier within the food
msreDesc	varchar(255)	a description of food portion, usually, amount consumable within a single meal
amount	double	how many portions defined by msreDesc are usually served or displayed at food label
gmWgt	double	How many grams are in serving. Is gmWgt is null, then food vendor published "gramless" food label with some msreDesc and nutrient values defined without specifying grams. <i>Gramless food has 1 serving with gmWgt null.</i>

3. FoodNutrientValue

Nutrient content of food

Field name	Type	Description
nutrNo	smallint	unique within a single food, refers to Nutrient.nutrNo
nutrVal	double	When food is gramless , then nutrVal contains absolute nutrient value in foods single serving, otherwise, for food with gram servings, nutrVal contains density of nutrient per 100g of food

4. Nutrient set

Note: Nutrient definitions and about 7,000 original foods are borrowed from USDA data set. See the documentation at <https://fdc.nal.usda.gov/>

Field name	Type	Description
nutrNo	smallint	primary key
nutrDesc	varchar(100)	Nutrient name
units	varchar(20)	Units of measure

nutrNo	nutrDesc	units
208	Calories	cals
204	Total Fat	g
606	Saturated Fat	g
605	Trans Fat	g
646	Polyunsaturated Fat	g
645	Monounsaturated Fat	g
601	Cholesterol	mg
307	Sodium	mg
205	Total Carbohydrates	g
291	Dietary Fiber	g
295	Soluble Fiber	g
269	Sugars	g
700	Sugar Alcohols	g

203	Protein	g
221	Alcohol Ethyl	g
262	Caffeine	mg
318	Vitamin A	iu
401	Vitamin C	mg
324	Vitamin D	iu
323	Vitamin E	mg
430	Vitamin K	mcg
404	Thiamin	mg
405	Riboflavin	mg
406	Niacin	mg
415	Vitamin B-6	mg
416	Biotin	mcg
417	Total Folate	mcg
418	Vitamin B-12	mcg
421	Choline	mg
410	Pantothenic Acid	mg
301	Calcium(Ca)	mg
303	Iron(Fe)	mg
306	Potassium(K)	mg
305	Phosphorus(P)	mg
304	Magnesium(Mg)	mg
309	Zinc(Zn)	mg
310	Chromium	mcg
314	Iodine	mcg
316	Molybdenum	mcg
317	Selenium(Se)	mcg
312	Copper(Cu)	mg
315	Manganese(Mn)	mg
901	Omega-3s	mg
851	Omega-3 ALA	mg
629	Omega-3 EPA	mg
621	Omega-3 DHA	mg
902	Omega-6s	mg

675	Omega-6 LA	mg
209	Starch	g
210	Sucrose	g
211	Glucose(Dextrose)	g
212	Fructose	g
213	Lactose	g
214	Maltose	g
321	Carotene Beta	mcg
322	Carotene Alpha	mcg
701	Added Sugars	g
245	Oxalate	mg
287	Galactose	g
297	Fiber insoluble	g
308	Sulfur	mg
311	Cobalt	mcg
313	Fluoride	mcg
319	Retinol	mcg
334	Beta-cryptoxanthin	mcg
337	Lycopene	mcg
338	Lutein+Zeaxanthin	mcg
341	Beta Tocopherol	mcg
342	Gamma Tocopherol	mcg
343	Delta Tocopherol	mcg
354	Boron	mcg
431	Folic Acid	mcg
435	Folate DFE	mcg
501	Tryptophan	mg
502	Threonine	mg
503	Isoleucine	mg
504	Leucine	mg
505	Lysine	mg
506	Methionine	mg
507	Cystine	mg
508	Phenylalanine	mg

509	Tyrosine	mg
510	Valine	mg
511	Arginine	mg
512	Histidine	mg
513	Alanine	mg
514	Aspartic acid	mg
515	Glutamic acid	mg
516	Glycine	mg
517	Proline	mg
518	Serine	mg
521	Hydroxyproline	mg
526	Cysteine	mg
528	Glutamine	mg
529	Taurine	mg
608	Caproic Acid	g
609	Caprylic Acid	g
610	Capric Acid	g
611	Lauric Acid	g
636	Phytosterol	mg
676	Erucic Acid	g
1000	Allulose	g
1001	CoQ10	mg
1002	Chloride	mg
1003	MCT	g
1004	Lutein	mcg
1005	Zeaxanthin	mcg

6. Data Types

Type name	Min value	Max value	Notes
int	-2147483648	2147483647	Integer number
mediumint	-8388608	8388607	Medium integer number
smallint	-32768	32767	Small integer number

double	4.9e-324	1.7e+308	up to 10-digits after decimal point
varchar(N)			variable length string, where N is max length
boolean			true or false

Java Client Example

package com.company.mynetdiary.services;

import java.io.*;

import java.net.*;

import java.util.*;

public class ApiClientExample {

public static void main(String[] args) **throws** Exception {

 String host = "www.mynetdiary.com";

 String partnerName = "****"; *//insert your value here*

 String partnerPassword = "****"; *//insert your value here*

final String urlBase = "://" + host + "/";

final String jsessionId = *partnerSignIn*(partnerName, partnerPassword, urlBase);

post("https", urlBase + "apiFindFoods.do", "beanInputString=milk", jsessionId);

post("https", urlBase + "apiFindFoods.do", "beanInputString=Silk+almond+milk", jsessionId);

post("https", urlBase + "apiFoodDetails.do", "foodId=1077", jsessionId);

post("https", urlBase + "apiUpcFindFood.do", "upc=787359177026", jsessionId);

 }

private static String *partnerSignIn*(String partnerName, String partnerPassword, String urlBase) **throws** IOException {

 URLConnection conn = *post*("https", urlBase + "apiPartnerSignin.do",

 "partnerName=" + partnerName + "&password=" + partnerPassword,

 null);

return *parseSessionId*(conn);

 }

```

private static URLConnection post(String protocol, String urlSuffix, String data, String
jsessionId) throws IOException {
    String url = protocol + urlSuffix;
    URLConnection conn = new URL(url).openConnection();
    if (jsessionId != null) {
        conn.setRequestProperty("Cookie", "JSESSIONID=" + jsessionId);
    }
    conn.setDoOutput(true);
    OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream());
    try {
        wr.write(data);
        wr.flush();

        System.out.println("----" + url + " request sent for "+data);

        BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
        try {
            String line;
            while ((line = rd.readLine()) != null) {
                System.out.println("----" + url + " response:");
                System.out.println(line);
            }
        } finally {
            rd.close();
        }
    } finally {
        wr.close();
    }

    return conn;
}

```

```

private static String parseSessionId(URLConnection conn) {
    String jsessionId = null;
    for (Map.Entry<String, List<String>> i : conn.getHeaderFields().entrySet()) {
        String key = i.getKey();
        if (key != null && key.equals("Set-Cookie")) {
            String v = i.getValue().toString();
            jsessionId = v.split("JSESSIONID=")[1];
            jsessionId = jsessionId.split(";")[0];
        }
    }
}

```

```

        System.out.println("Parsed JSESSIONID=" + jsessionId);
        break;
    }
}
return jsessionId;
}
}

```

Connecting over SSL

When the Food Search API is called over SSL, a proper certificate chain should exist in your certificate store.

MyNetDiary's SSL certificate is issued by GoDaddy, which requires having "Go Daddy Root Certificate Authority - G2" and "Go Daddy Secure Certificate Authority - G2" certificates in your certificate store.

If you don't have these certificates in your certificate store, you can download them as `gd_bundle-g2.crt` available at <https://certs.godaddy.com/repository/> and install following your development tools infrastructure. For Java, this means adding to your "keystore" with Java "keytool" commands.

Using Certificates with Java

The recommended practice is to use a special keystore for the webservice client code, so that it can be deployed with the client build and does not depend on system or JDK-provided keystore. If you use such special keystore, you will need to specify its location, password, etc. as JVM system properties (java -D..., e.g. -Djavax.net.ssl.keyStore), or set in code with `System.setProperty`. The following StackOverflow question provides a good "how-to" description: <https://stackoverflow.com/questions/5871279/java-ssl-and-cert-keystore>