MyNetDiary Food Search API v2.1

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 http://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 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 HTML class name for spans marking found keywords in returned food descriptions

NOTE: Contributed foods are not found by API

Response

Please see Data Dictionary for field descriptions
HTTP Status 200
JSON example for beanInputString=milk:


```
{
  "ms": 20,
  "numberOfAvailableEntries": 17226,
  "entries": [{
    "imageId": 101,
    "gramlessAmountMeasure": "200 ml",
    "beanId": 9542412,
    "descForUi": "Whole milk by tesco",
    "isGramless": true,
```

```
    "contrib": false
}, {
    "imageId": 101,
    "gramlessAmountMeasure": "236 ml",
    "beanId": 9577508,
    "descForUi": "2% reduced fat milk by schneider's",
    "isGramless": true,
    "contrib": false
}, {
    "imageId": 101,
    "gramlessAmountMeasure": "250 ml",
    "beanId": 10662252,
    "descForUi": "1% partly skimmed milk by dairyland",
    "isGramless": true,
    "contrib": false
}, {
    "imageId": 101,
    "gramlessAmountMeasure": "240 ml",
    "beanId": 10930087,
    "descForUi": "1% lowfat milk by darigold",
    "isGramless": true,
    "contrib": false
}, {
    "imageId": 101,
    "gramlessAmountMeasure": "236 ml",
    "beanId": 9327110,
    "descForUi": "Low fat 1% milk by hollandia dairy",
    "isGramless": true,
    "contrib": false
}, {
    "imageId": 101,
    "gramlessAmountMeasure": "240 ml",
    "beanId": 9590109,
    "descForUi": "1% milkfat lowfat milk by winn dixie",
    "isGramless": true,
    "contrib": false
}, {
    "imageId": 101,
    "gramlessAmountMeasure": "240 ml",
    "beanId": 9021879,
    "descForUi": "0.5% milkfat low fat milk by giant eagle",
    "isGramless": true,
    "contrib": false
}, {
    "imageId": 101,
    "gramlessAmountMeasure": "240 ml",
    "beanId": 9875053,
    "descForUi": "1% milkfat low fat milk by hill country fare",
```

```
    "isGramless": true,
    "contrib": false
  }, {
    "imageId": 107,
    "beanId": 1026,
    "descForUi": "Cheese mozzarella whole milk",
    "contrib": false
  }, {
    "imageId": 101,
    "gramlessAmountMeasure": "236 ml",
    "beanId": 9327109,
    "descForUi": "Milk 1% milkfat vitamins a & d grade pasteurized homogenized by hollandia dairy",
    "isGramless": true,
    "contrib": false
  }]
}
```


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",
  "nutrients" : [
    {"val":60,"nutr":208},
    {"val":3.25,"nutr":204},
    {"val":4.52,"nutr":205},
    {"val":3.22,"nutr":203},
    {"val":1.865,"nutr":606},
    {"val":0.195,"nutr":646},
    {"val":0.812,"nutr":645},
    {"val":10,"nutr":601},
    {"val":40,"nutr":307},
    {"val":0,"nutr":291},
    {"val":5.26,"nutr":269},
```

```
      {"val":102,"nutr":318},
      {"val":0,"nutr":401},
      {"val":113,"nutr":301},
      {"val":0.03,"nutr":303},
      {"val":0.036,"nutr":415},
      {"val":0.44,"nutr":418},
      {"val":40,"nutr":324},
      {"val":0.06,"nutr":323},
      {"val":0.2,"nutr":430},
      {"val":10,"nutr":304},
      {"val":91,"nutr":305},
      {"val":143,"nutr":306},
      {"val":0.4,"nutr":309},
      {"val":0.011,"nutr":312},
      {"val":0.003,"nutr":315},
      {"val":3.7,"nutr":317},
      {"val":0.044,"nutr":404},
      {"val":0.183,"nutr":405},
      {"val":0.107,"nutr":406},
      {"val":0.362,"nutr":410},
      {"val":5,"nutr":417},
      {"val":0,"nutr":221},
      {"val":0,"nutr":262},
      {"val":88.32,"nutr":255},
      {"val":0,"nutr":210},
      {"val":0,"nutr":211},
      {"val":0,"nutr":212},
      {"val":5.26,"nutr":213},
      {"val":0,"nutr":214},
      {"val":5,"nutr":321},
      {"val":0,"nutr":322}
    ],"weights":
      [
      {"sortOrder":1,"msreDesc":"cup","gmWgt":"244."},
      {"sortOrder":2,"msreDesc":"tbsp","gmWgt":"15."},
      {"sortOrder":3,"msreDesc":"fl oz","gmWgt":"30.5"},
      {"sortOrder":4,"msreDesc":"quart","gmWgt":"976."},
      {"sortOrder":987,"msreDesc":"oz","gmWgt":"28.35"},
      {"sortOrder":988,"msreDesc":"gram","gmWgt":"1."}
    ]
}
```


UPC lookup

Request

URL: **apiUpcFindFood.do**
HTTP header named Cookie, header value is formatted as “JSESSIONID=...”

Parameter:
upc - food upc code

NOTE: Contributed foods are not found by API

Response

Please see Data Dictionary for field descriptions
HTTP Status 200
JSON example for upc=36800180253:


```
{
  "beanId": 1077,
  "beanDesc": "Milk whole 3.25% milkfat",
  "weights": [
    {
      "amountId": "1",
      "itemDesc": "cup = 149kcal in 244g",
      "amountMsreDesc": "cup",
      "msreDesc": "cup"
    }, {
      "amountId": "2",
      "itemDesc": "tbsp = 9kcal in 15g",
      "amountMsreDesc": "tbsp",
      "msreDesc": "tbsp"
    }, {
      "amountId": "3",
      "itemDesc": "fl oz = 19kcal in 31g",
      "amountMsreDesc": "fl oz",
      "msreDesc": "fl oz"
    }, {
      "amountId": "4",
      "itemDesc": "quart = 595kcal in 976g",
      "amountMsreDesc": "quart",
      "msreDesc": "quart"
    }, {
      "amountId": "987",
      "itemDesc": "oz = 17kcal in 28g",
      "amountMsreDesc": "oz",
      "msreDesc": "oz"
    }, {
      "amountId": "988",
      "itemDesc": "gram = 1kcal in 1g",
      "amountMsreDesc": "gram",
      "msreDesc": "gram"
    }, {
      "amountId": "1011",
      "itemDesc": "ml = 1kcal in 1g",
```

```
        "amountMsreDesc": "ml",
        "msreDesc": "ml"
      }, {
        "amountId": "1013",
        "itemDesc": "teaspoon = 3kcal in 5.1g",
        "amountMsreDesc": "teaspoon",
        "msreDesc": "teaspoon"
      }],

  "foodLabel": {
    "TotalCarbsPercent": "4%",
    "Calcium": "28%",
    "VitaminC": "0%",
    "Calories": "149",
    "Chol": "24mg",
    "SaturatedFat": "4.6g",
    "TransFat": "",
    "SaturatedFatPercent": "23%",
    "FoodName": "Milk whole 3.25% milkfat",
    "Sugars": "12g",
    "FatPercent": "12%",
    "TotalCarbs": "12g",
    "UnsatFatMono": "2g",
    "FiberPercent": "",
    "VitaminA": "8%",
    "TotalFat": "7.9g",
    "custom": false,
    "Protein": "7.7g",
    "CholPercent": "8%",
    "CaloriesFromFat": "71",
    "Serving": "cup (244g)",
    "i": 101,
    "SugarAlcohol": "0g",
    "Sodium": "105mg",
    "UnsatFatPoly": "0.5g",
    "contrib": false,
    "Iron": "0%",
    "Fiber": ""
  }
}
```


Data Dictionary

1. Food

Food header information

Field name	Type	Description
foodId	int	primary key, food identifier
foodDesc	varchar(255)	Food description

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. *Gramless food has 1 serving with gmWgt null*.

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 http://www.nal.usda.gov/fnic/foodcomp/search/

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

The list of 44 nutrients existing in MyNetDiary database is provided below.
Notice, that 36 nutrients have researchType=2 and 8 nutrients have researchType=1
researchType=2 means that nutrient values are entered in MyNetDiary dataset for all foods, including original USDA foods and food labels published by food vendors.
researchType=1 means that nutrient values are entered in MyNetDiary dataset only for foods inherited from USDA dataset (about 7,000 foods). MyNetDiary dataset does not contain values of nutrients with researchType=1 for vendor (non-USDA) foods.

nutrNo	nutrDesc	units	researchType
208	Calories	kcal	2
204	Total Fat	g	2
605	Trans Fat	g	2
606	Saturated Fat	g	2
646	Polyunsaturated Fat	g	2
645	Monounsaturated Fat	g	2
601	Cholesterol	mg	2
307	Sodium	mg	2
205	Total Carbohydrates	g	2
291	Dietary Fiber	g	2
269	Sugars	g	2
203	Protein	g	2
318	Vitamin A	iu	2
401	Vitamin C	mg	2
301	Calcium(Ca)	mg	2

303	Iron(Fe)	mg	2
415	Vitamin B-6	mg	2
418	Vitamin B-12	mcg	2
324	Vitamin D	iu	2
323	Vitamin E	mg	2
430	Vitamin K	mcg	2
304	Magnesium(Mg)	mg	2
305	Phosphorus(P)	mg	2
306	Potassium(K)	mg	2
309	Zinc(Zn)	mg	2
312	Copper(Cu)	mg	2
315	Manganese(Mn)	mg	2
317	Selenium(Se)	mcg	2
404	Thiamin	mg	2
405	Riboflavin	mg	2
406	Niacin	mg	2
410	Pantothenic Acid	mg	2
417	Total Folate	mcg	2
221	Alcohol Ethyl	g	2
262	Caffeine	mg	2
209	Starch	g	2
255	Water	g	1
210	Sucrose	g	1
211	Glucose(Dextrose)	g	1
212	Fructose	g	1
213	Lactose	g	1
214	Maltose	g	1
321	Carotene Beta	mcg	1
322	Carotene Alpha	mcg	1

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)	see notes		variable length string, where N is max length

Java Client Example

```
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("http", urlBase + "apiFindFoods.do", "beanInputString=milk", jsessionId);

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

    post("http", urlBase + "apiUpcFindFood.do", "upc=71840202046", 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");

            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("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 your 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: http://stackoverflow.com/questions/5871279/java-ssl-and-cert-keystore