Verti Open data API interface

Table of Contents

1 General description

Verti open data api interface can be used to download statistical table data and metadata from a Verti database. HTTP GET or POST requests are used for data downloading and query parameters are specified in the query string as parameter1=value1&parameter2=value2& and so on.

2 Available formats

Verti Open data interface queries return data in the following formats depending on the konv parameter setting

- CSV
- JSON
- JSON-STAT
- XML

CSV, JSON, JSON-STAT and XML return text data and they use UTF-8 character set. UTF-8 BOM (Byte Order Mark) is added to the beginning of the response so that the programs reading the data know how to show for example Scandinavian characters correctly.

2.1 Comma-separated values (csv)

This format is used for all levels: stats, dims, class and data

Semicolon is used as field separator, string values are enclosed with double quote characters, number values are not enclosed..

Example:

```
"D1";"Alue2";"N";320;"T"
"D1";"- Kunnat aakkosjärjestyksessä 2013";"N";321;"T"
"D1";"- Seutukunnat 2013";"N";73;"T"
"D1";"- Maakunnat 2013";"N";22;"T"
"D1";"- Elinkeino-, liikenne- ja ympäristökeskukset 2013";"N";19;"T"
"D2";"Vuosi";"N";12;"T"
"D3";"Aika";"N";13;"T"
"D4";"Toimiala (TOL 2008)";"N";9;"T"
"D5";"Yritystiedot";"N";2;"T"
```

2.2 JSON

This format is used for all levels: stats, dims, class and data

For more information about the JSON format see http://json.org/

Example:


```
{"dimension":[
{"id":"D1","classification":[
{"label":"Alue2","elim":"N","size":320,"show":"T"},
{"label":"- Kunnat aakkosjärjestyksessä 2013","elim":"N","size":321,"show":"T"},
{"label":"- Seutukunnat 2013","elim":"N","size":73,"show":"T"},
{"label":"- Maakunnat 2013","elim":"N","size":22,"show":"T"},
{"label":"- Elinkeino-, liikenne- ja ympäristökeskukset 2013","elim":"N","size":19,"show":"T"}
]},
{"id":"D2","classification":[
{"label":"Vuosi","elim":"N","size":12,"show":"T"}
]},
{"id":"D3","classification":[
{"label":"Aika","elim":"N","size":13,"show":"T"}
]},
{"id":"D4","classification":[
{"label":"Toimiala (TOL 2008)","elim":"N","size":9,"show":"T"}
]},
{"id":"D5","classification":[
{"label":"Yritystiedot","elim":"N","size":2,"show":"T"}
]}
]}
```


2.3 JSON-STAT

This format is used for class and data levels only

For more information about the JSON-STAT format see http://json-stat.org/

2.4 XML

This format is used for all levels: stats, dims, class and data

Example


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Data>
<r i="1"><c i="dim"><t>D1</t></c><c i="class"><t>Alue2</t></c><c i="elim"><t>N</t></c><c
i="size"><v>320</v></c><c i="show"><t>T</t></c></r>
<r i="2"><c i="dim"><t>D1</t></c><c i="class"><t>- Kunnat aakkosjärjestyksessä 2013</t></c><c
i="elim"><t>N</t></c><c i="size"><v>321</v></c><c i="show"><t>T</t></c></r>
<r i="3"><c i="dim"><t>D1</t></c><c i="class"><t>- Seutukunnat 2013</t></c><c i="elim"><t>N</t></c><c
i="size"><v>73</v></c><c i="show"><t>T</t></c></r>
```

<r i="4"><c i="dim"><t>D1</t></c><c i="class"><t>- Maakunnat 2013</t></c><c i="elim"><t>N</t></c><c i="size"><v>22</v></c><c i="show"><t>T</t></c></r>
<r i="5"><c i="dim"><t>D1</t></c><c i="class"><t>- Elinkeino-, liikenne- ja ympäristökeskukset 2013</t></c><c i="elim"><t>N</t></c><c i="size"><v>19</v></c><c i="show"><t>T</t></c></r>
<r i="6"><c i="dim"><t>D2</t></c><c i="class"><t>Vuosi</t></c><c i="elim"><t>N</t></c><c i="size"><v>12</v></c><c i="show"><t>T</t></c></r>
<r i="7"><c i="dim"><t>D3</t></c><c i="class"><t>Aika</t></c><c i="elim"><t>N</t></c><c i="size"><v>13</v></c><c i="show"><t>T</t></c></r>
<r i="8"><c i="dim"><t>D4</t></c><c i="class"><t>Toimiala (TOL 2008)</t></c><c i="elim"><t>N</t></c><c i="size"><v>9</v></c><c i="show"><t>T</t></c></r>
<r i="9"><c i="dim"><t>D5</t></c><c i="class"><t>Yritystiedot</t></c><c i="elim"><t>N</t></c><c i="size"><v>2</v></c><c i="show"><t>T</t></c></r>
</Data>

3 Interface levels

There are four different interface levels for queries

- Stats
- Dims
- Class
- Data

Stats, Dims and Class levels are used to return metadata about the statistics tables and Data level is used to return the actual data.

3.1 Statistics

The uppermost level returns the names of the statistics files present in the Verti database.

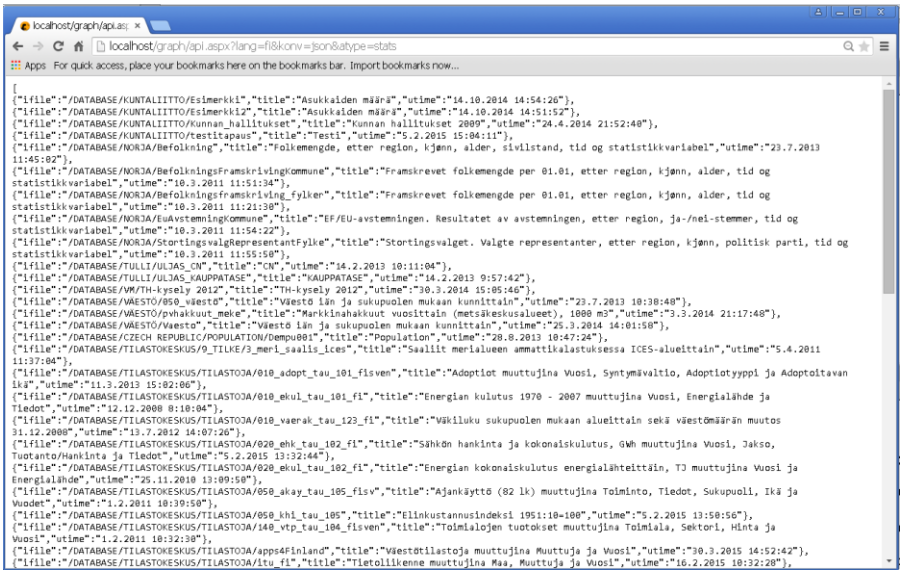
Query parameter settings : lang=<language code> atype=stats konv=<output format> find=<keyword>. Find parameter is optional, others parameters are mandatory

Available output formats are csv, json, xml.

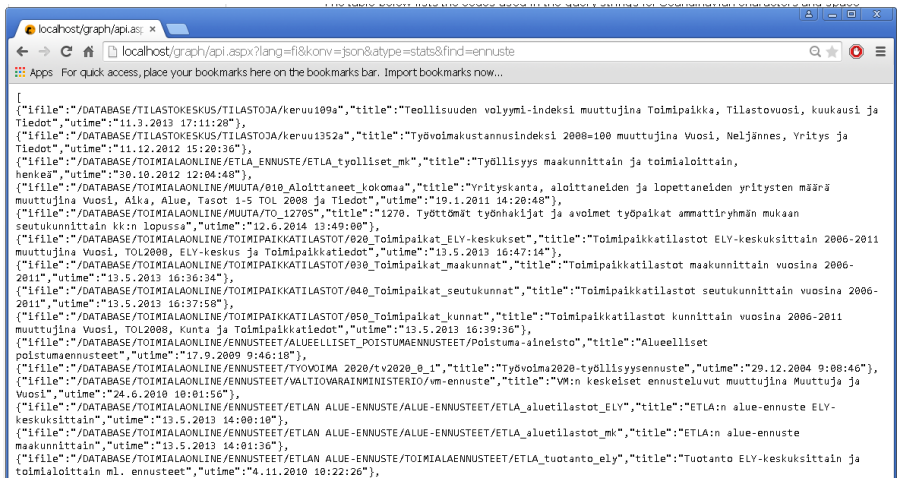
Query output has the following data fields

Data field name	Explanation
ifile	The name of the statistics file with relative data directory path. These names can be used with the ifile query parameter in dims, class and data queries.
title	The title of the statistics
utime	Last update time of the statistics, timestamp format is DD.MM.YYYY HH24:MI:SS

Statistics query example with json format output:



Using find parameter with statistics query to return only those statistics that have word “ennuste” in the descriptions.



3.2 Dimensions

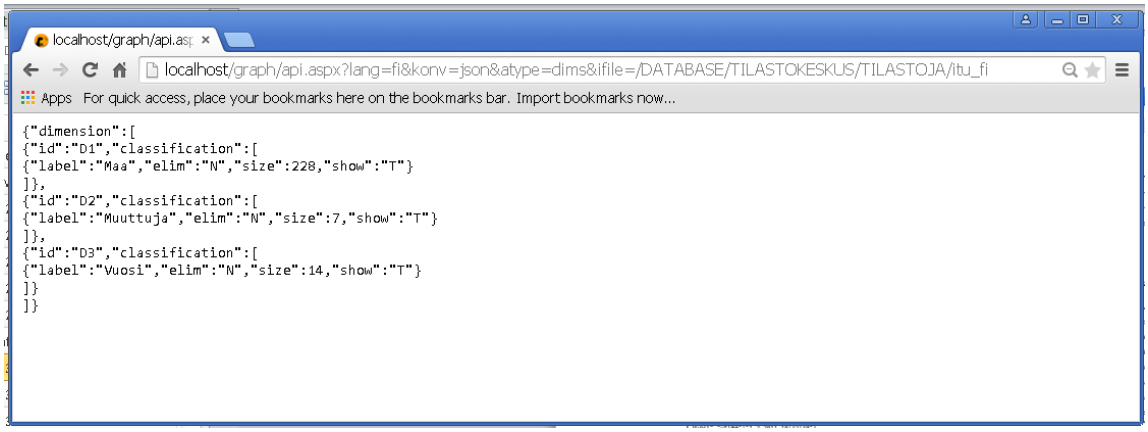
This level returns the dimensions that are available for a certain statistics file specified with ifile parameter. Stats query can be used to fetch the names of the statistics files that are available.

Query parameter settings : lang=<language code> atype=dims konv=<output format> ifile=<statistics file name>

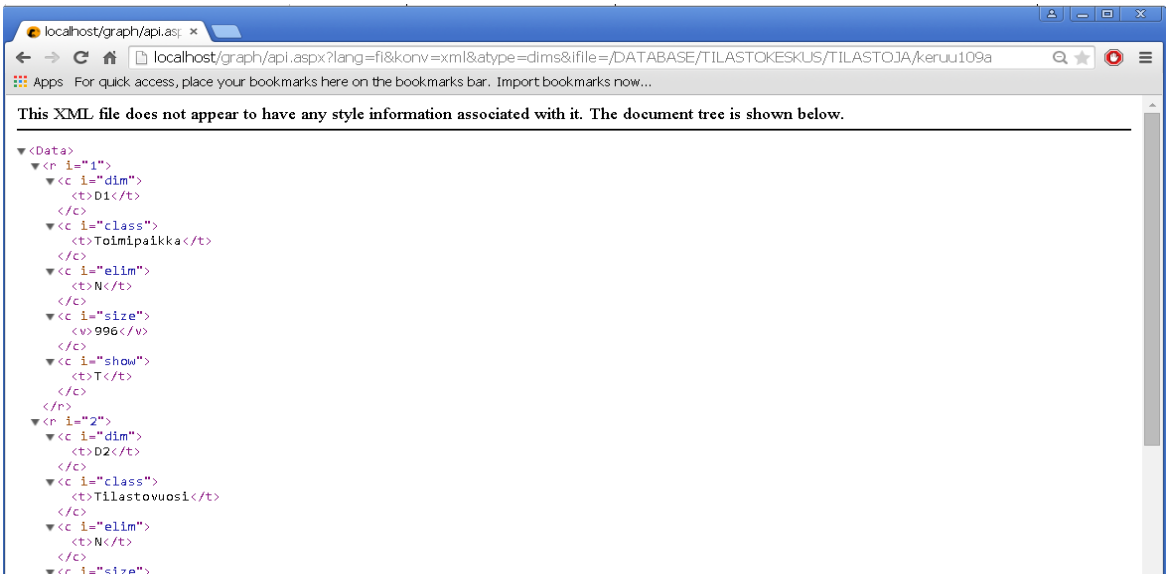
Available output formats are csv, json, xml.

Data field name	Explanation
id	Dimension id has the following formats: D<N> when normal dimension and N is dimension index. The same value can be repeated in order. This means that those classifier-rows are aggregates in this dimension. In data-query you can select only from one classifier per dimension. V<N> when at issue dimension is variable-dimension. This must always be the last one in data-query selections. In many cases there isn't separately named variable dimension.
label	Classifier identifier, this value can be used with class parameter in class query
elim	N when classifier (or merely dimension) is not able to eliminate (= no elimination rule set) and then you must make selection from one classifier in dimension. Y when classifier (or merely dimension) can be eliminated (= there is elimination rule set, which is very often "grand total"). In this case you don't have make selection from this dimension in data-query.
size	The number of values in this classification.
show	T when default output for classifier is class-text CT when default output for classifier is class-code + ' ' + class-text C when default output for classifier is class-code

Example dimensions query with json format output



Dimensions query with XML output:



3.3 Classifications

This level returns the values that are in the classification (specified with class parameter) for a certain statistics file (specified with ifile parameter). Dims query can be used to fetch the classifications that are available for a certain statistics file.

Query parameter settings : lang=<language code> atype=class konv=<output format> ifile=<statistics file name> class=<classification name>

If class parameter is omitted or it has empty value and json output format is used then the query will return all the classifications and their values defined for the statistics ifile.

Available output formats are csv, json, json-stat, xml.

Data field name	Explanation
code	Classification code
text	Text that corresponds the code

Class query example with json format output

3.4 Data

This level returns the data values that are returned with certain class value combinations from a certain statistics file (specified with ifile parameter). Class query can be used to fetch the classification values that are used as query parameters.

Query parameter settings : lang=<language code> atype=data konv=<output format> ifile=<statistics file name> select=codes

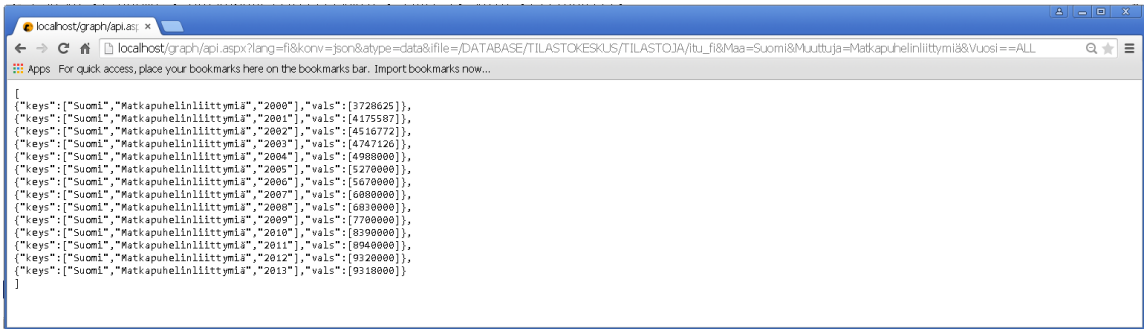
Parameter name	Parameter values with explanations
Select	Codes if you want class-codes in response and not those defaults which are seen in dims-query. This also affects to classifier selections below when value-based.
Exclude	ZRS when you want exclude records where all vals are zeros. NaN when you want exclude records where all vals are missing or hidden values ZAN when you want exclude records where all vals are ZRS or NaN
<classifier>	<value> where value is given in class output format (T, CT or C) <value list> where each value is given like above. In the response the order is the same you give here. The delimiter character is “ (straight double quote U+0022) in normal query, because comma can be in class-code or class-text. And therefore it cannot be used. =ALL when you want all class rows =ALLBUT <N> when you want all rows but exclude N rows from the start or from the end in negative value = FIRST <N> when you want first N rows from the start of classier and if N is list of count-values, then second count-value is excluded and third count included and so on. = LAST <N> when you want last N rows from the end of classier and if N is list of count-values, then like in above. =GEN <value-start> when you want generic start of code (text)
<classifier>	Next classifier which you want or you have included (has elim=N). The order of classifier selections is free, but the last one must from variable dimension if there exists.

Available output formats are csv, json, json-stat, xml.

Data field name	Explanation
Keys	List of classifier output values (as defined)
Values	List of each variable selected in numeric mode even when variable dimension doesn't exist there is 1 variable. If variable item is missing or is hidden, then there is in text-mode one of following: ".", ".." or "…".

3.4.1 Data query examples

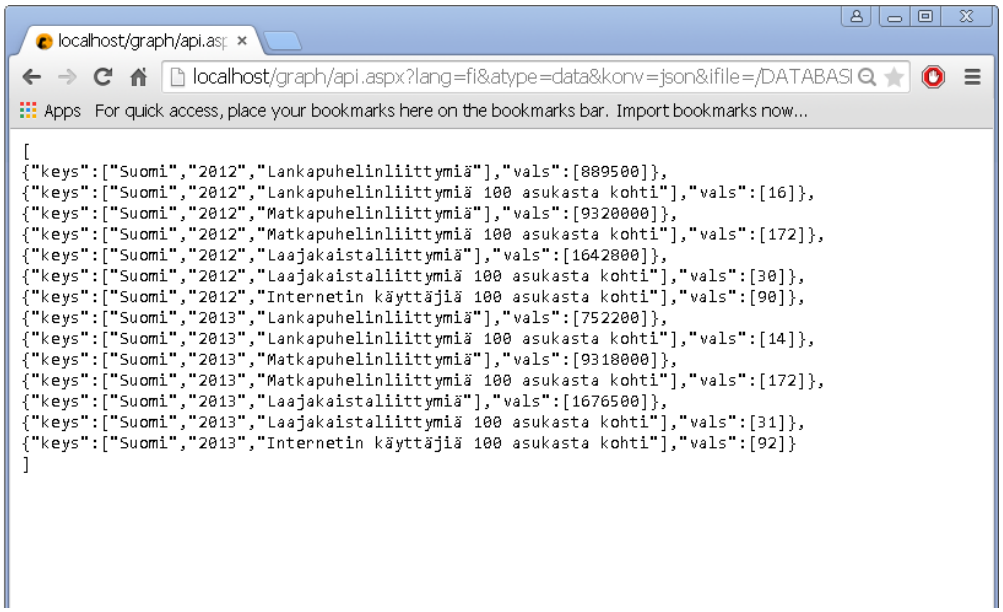
Data query example with JSON format



Same query with JSON-STAT format:

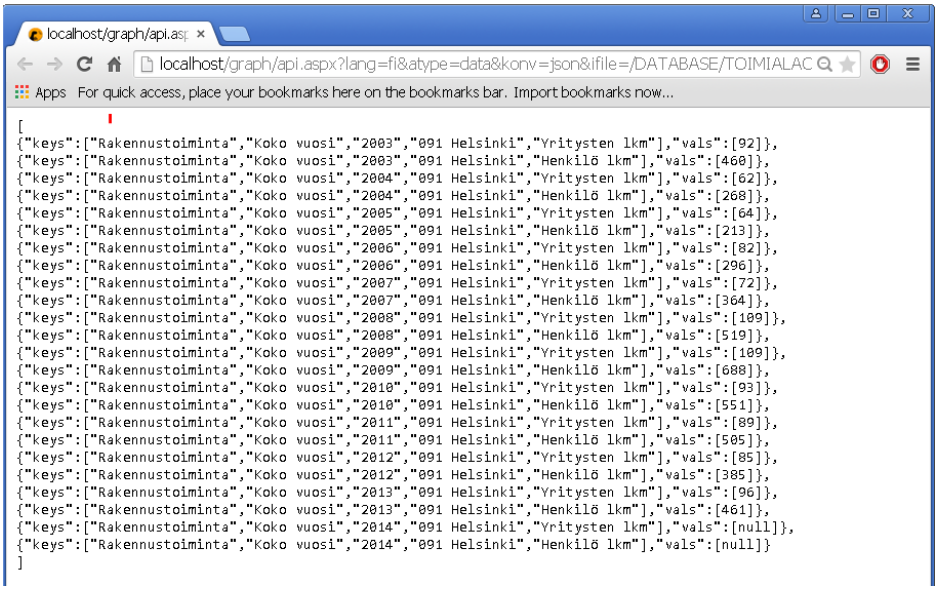
The following data query uses LAST parameter setting to return only the last 2 values of Vuosi class:

http://localhost/graph/api.aspx?lang=fi&atype=data&konv=json&ifile=/DATABASE/TILASTOKESKUS/TILASTOJA/itu_fi&Maa=246&Vuosi==LAST*;2&Muuttuja==ALL



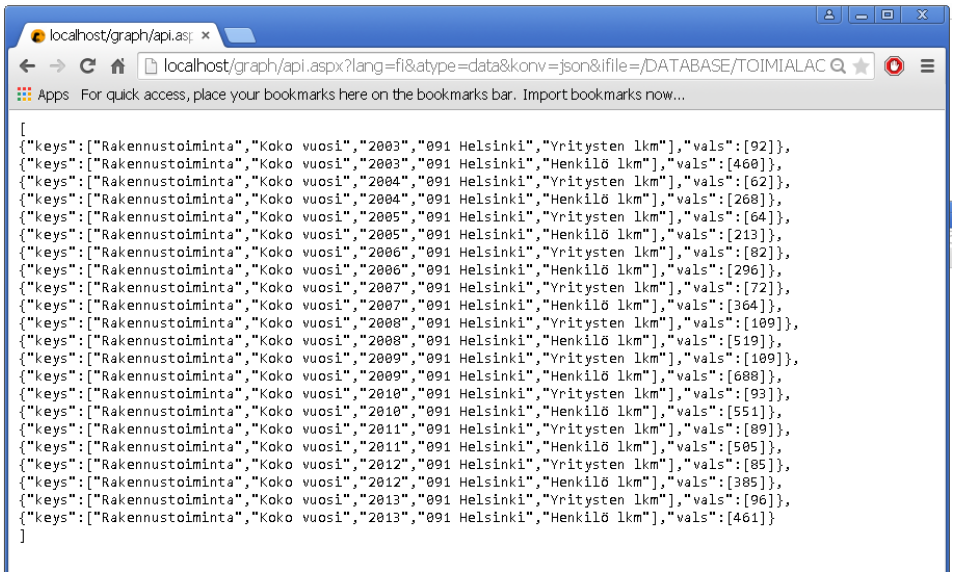
The following example data query returns some null values

http://localhost/graph/api.aspx?lang=fi&atype=data&konv=json&ifile=/DATABASE/TOIMIALAONLINE/KONKURSSIT/Konkurssit_alueittain&Toimiala*;%28TOL*;2008%29=F&Aika=SSS&Vuosi==ALL&Alue2=091&Yritystiedot==ALL



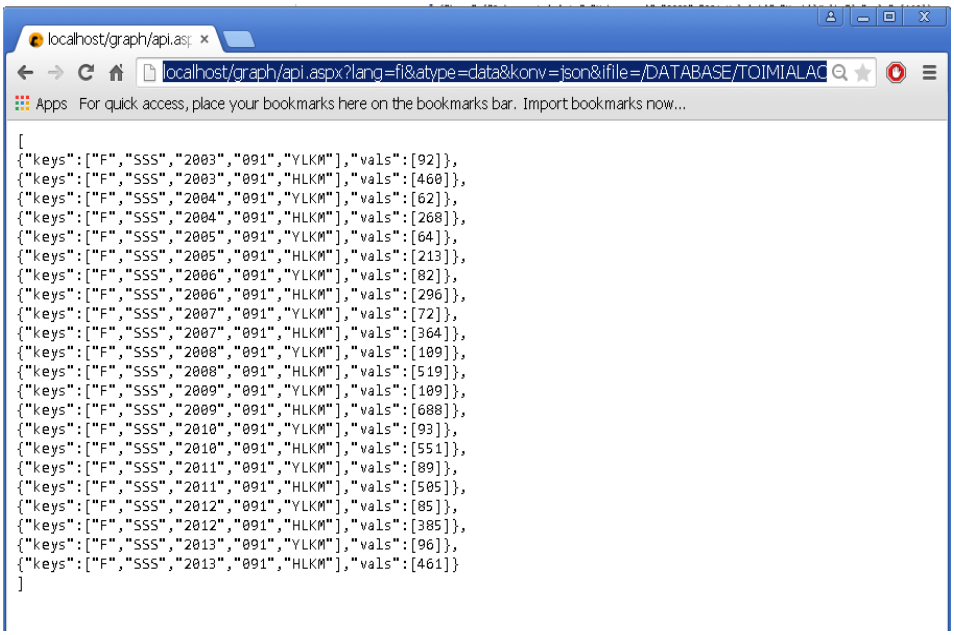
These values can be removed from the result set by using the Exclude parameter

http://localhost/graph/api.aspx?lang=fi&atype=data&konv=json&ifile=/DATABASE/TOIMIALAONLINE/KONKURSSIT/Konkurssit_alueittain&Toimiala*;%28TOL*;2008%29=F&Aika=SSS&Vuosi==ALL&Alue2=091&Yritystiedot==ALL&Exclude=ZAN



If Select=Codes parameter setting is used the query output will alter as follows:

http://localhost/graph/api.aspx?lang=fi&atype=data&konv=json&ifile=/DATABASE/TOIMIALAONLINE/KONKURSSIT/Konkurssit_alueittain&Select=Codes&Toimiala*;%28TOL*;2008%29=F&Aika=SSS&Vuosi==ALL&Alue2=091&Yritystiedot==ALL&Exclude=ZAN



4 Data handlers

Data handlers can be used to return the data in a specific format (CSV, JSON, JSON_STAT, or XML).by replacing api.aspx with api.csv, api.json, or api.xml in the query.

Conventional query using api.aspx	Equivalent query using data handler
http://localhost/graph/api.aspx?lang=fi&atype=stats&konv=json	http://localhost/graph/api.json?lang=fi&atype=stats
http://localhost/graph/api.aspx?lang=fi&atype=stats&konv=json-stat	http://localhost/graph/api.json?lang=fi&atype=stats&konv=json-stat
http://localhost/graph/api.aspx?lang=fi&atype=stats&konv=csv	http://localhost/graph/api.csv?lang=fi&atype=stats
http://localhost/graph/api.aspx?lang=fi&atype=stats&konv=xml	http://localhost/graph/api.xml?lang=fi&atype=stats

If a data handler is used then the konv parameter can be omitted from the query. The only exception is json-stat data handler where konv=json-stat parameter value must be specified to inform the json handler to return data using json-stat format.

Generally speaking the data handler always overrides the konv parameter setting.

Data handlers are defined in **Graph\web.config** file with the following entries:

In <httpHandlers> section these entries must be present

```
<add verb="*" path="Api.json" type="Graph.Handlers.JsonApiHandler,Graph.Handlers" />
<add verb="*" path="Api.csv" type="Graph.Handlers.CsvApiHandler,Graph.Handlers" />
<add verb="*" path="Api.xml" type="Graph.Handlers.XmlApiHandler,Graph.Handlers" />
<add verb="*" path="Api.xlsx" type="Graph.Handlers.XlsxApiHandler,Graph.Handlers" />
```

In <system.webServer> <handlers> section these entries must be present

```
<add name="JsonApiHandler" path="Api.json" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode,runtimeVersionv4.0,bitness32" />
```

```
<add name="CsvApiHandler" path="Api.csv" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode,runtimeVersionv4.0,bitness32" />
<add name="XmlApiHandler" path="Api.xml" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode,runtimeVersionv4.0,bitness32" />
<add name="XlsxApiHandler" path="Api.xlsx" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode,runtimeVersionv4.0,bitness32" />
```

In these entries *scriptProcessor* .NET Framework version and *preCondition runtimeVersion* and *bitness* values can vary. They must be updated to use the .NET Framework version that has been installed in the machine and also bitness must be set to bitness32 or bitness64 depending on the bitness of the Windows version in use.

5 HTTP POST requests

HTTP POST requests can be used instead of the HTTP GET request. The only difference between them is that the query parameters are now specified in the message body instead of them being part of the URL string.

Following example shows how the parameters are specified in a HTTP POST request message

```
POST http://localhost/graph/api.aspx HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:38.0) Gecko/20100101 Firefox/38.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fi,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 109
Cookie: ASP.NET_SessionId=0fm35ulghjk54k3oqehrcqcl
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

lang=fi&konv=xml&atype=class&ifile=%2FDATABASE%2FTOIMIALAONLINE%2FMUUTA%2F010_Aloittaneet
_kokomaa&class=Vuosi
```

6 Special characters in queries

The table below lists the codes used in the query strings for Scandinavian characters and space character:

ä	***228;**
ö	***246;**
å	***229;**
Ä	***196;**
Ö	***214;**
Å	***197;**
<space>	***;**

For example the word “Väestö” would be coded like this in the query string:

http://localhost/graph/api.aspx?lang=fi&konv=json&atype=stats&find=V*228;est*246;

7 Using Verti Open Data API

Following small C# program shows how Verti Open Data API can be used to get data. This particular program lists all the classifications that are available in a Verti database.

```
void Main()
{
    using (WebClient client = new WebClient())
    {

        // Download data.
        string data = client.DownloadString(@"http://localhost/graph/api.aspx?lang=fi&konv=csv&atype=stats");
        string[] stringSeparators = new string[] {"\r\n"};
        string[] lines = data.Split(stringSeparators, StringSplitOptions.None);
        // Write values.
        Console.WriteLine("--- WebClient result ---");
        // Console.WriteLine(lines[0]);

        string resdata ="";
        string[] fieldSeps = new string[] {";"};

        for (int i=0; i < lines.Length; i++){

        string[] fields = lines[i].Split(fieldSeps, StringSplitOptions.None);
        string ifile = fields[0].Replace(@"""","").Trim();
        if (!String.IsNullOrEmpty(ifile)){
        ifile = @"http://localhost/graph/api.aspx?lang=fi&konv=json&atype=class&ifile=" + ifile;
        Console.WriteLine(ifile);
        try {
        resdata = client.DownloadString(ifile);
        Console.WriteLine(resdata);
        } catch (Exception e) {Console.WriteLine("*** ERROR *** " + e.Message);}
        }
        }
}
}
// Define other methods and classes here
```

The example program produces this kind of output:

```
--- WebClient result ---
http://localhost/graph/api.aspx?lang=fi&konv=json&atype=class&ifile=/DATABASE/KUNTALIITTO/Esimerkki
{"classification":[
{"id":"D1","label":"Vuosi",
"class":[
{"code":"2005","text":"2005"},
{"code":"2006","text":"2006"}
]},
{"id":"D2","label":"KuntaLista",
"class":[
{"code":"004","text":"Alahärmä"},
{"code":"005","text":"Alajärvi"},
{"code":"006","text":"Alastaro"},
{"code":"009","text":"Alavieska"},
{"code":"010","text":"Alavus"}
]},
```