APALOJS is a JavaScript web service that provides access to data in a separate Jedox or Palo server. Unlike the Apalo/Jedox http api, proper cube, dimension, and element names are used. It is also optimized to return tables of data in a single server call.

Also, a single APALOJS connection to Apalo or Jedox provides shared support for multiple users. Typically, a small number of connections (e.g. 3) are used in a pooled manner.

APALOJS is directly compatible with client-side Javascript applications, using JSON formats for data transport.

APALOJS is a Node.JS Express application.

APALOJS supports the following requests:

**/apalo/data**

GET one or a range of cell values from a cube.

PUT one or a range of cell values to a cube.

**/apalo/elements**

GET element names in a dimension.

**/apalo/table**

GET a table of values using either dimension parent-child expansion or a preconfigured definition file. This allows a large range of data to be retrieved in a single server call.

**Usage for GET /apalo/data**

Query Request:

GET /apalo/data?db="Database name"&cube="Cube name",dims=Dimension Values

“dims” is a sequential list of dimension values in the order they are defined in the cube, semicolon separated and quoted as needed.

The return value is an array in JSON, i.e. square brackets ([...]), or an error message string (if not in square brackets and with a status other than 200).

One or more of the dimensions can be a comma-separated list or a range specification or a combination of these. Use a leading "=" in this case. Ranges only work for numeric element names. E.g. you might have an end of month element and also a range of days:

="31 MTD",1-31

Another example: To retrieve both "Budget" and "Actual" in a dimension, you would use:

="Budget","Actual"

When multiple ranges are supplied, the returned array is the cartesian product of the input ranges (rightmost element varies first).

**Usage for GET /apalo/elements**

Query Request:

GET /apalo/elements?db=…&dims=…&parent=…&type=…&format=…&levels=…

db = Database name

dim = dimension name(s)

parent = parent element(s), or empty for root

type = Base, Cons, or All (default) - only 1st letter checked

format = Hierarchy or Flat (default) - only 1st letter checked

levels = # of levels to return (1 is 1st children of parent), 0 for all (default)

Values are returned in a JSON array. Each element is either:

1. A base element name, or all elements in a flat view.
2. An array with a consolidated element in index 0 and a list of children in subsequent positions. Note that if there are multiple levels present, the children could be arrays with their own children.

Multiple requests can be batched. In this case, each of the query parameters after the database name should be lists of values enclosed in square brackets. In this case, the return value is an array of value arrays as described above. For example:

&dim=[Company,Year]&format=[H,F]&type=[A,B]

**Usage for GET /apalo/table**

This request retrieves a table in one of two formats. It can either expand a dimension to return a parent and its children in additional rows or it can read pre-defined dimension values from a definition file stored on the server.

Common query parameters for both forms of the GET request are:

db = Database name

cube = Cube name

dims = list of dimension values, semicolon separated and quoted as needed

headers = add elements header row, this is string put at front if row descriptions are used (if not, the value of this string is ignored).

The return value is an array of arrays, each sub-array representing one row in the expansion or one line in the definition file . This is JSON-compatible formatting.

See "GET /apalo/data" for additional information.

**Query Format #1:**

expand = Dimension name to expand. The parent element is in the "dims" list.

level = Number of levels to expand

indent = Number of spaces to add in descriptions for each level

nonempty = 1/true to supress empty/zero rows (except for first)

numberformat = numeric format specification. First character is # of decimal places. Additional characters following are:

"$" to format as currency

"%" to format as percentage

"," to use 1000's separators

"(" to format negative numbers in parenthesis

Note: When this formatting is used, converted string values are returned instead of numbers.

Here is a sample from the bikersmgmtreport.js file:

http://localhost:3000/apalo/table?expand=Products&level=1&indent=1&db="Biker"&cube="Orders2"&numberformat="0,"&dims="2013";="May","May YTD";"All Products";"All Customers";"All Channels";="Actual","Budget";="Units","Sales"

**Query Format #2:**

form = name of local definition file (without the .txt extension)

The definition file has a list of dimension elements on each line. These correspond to variable values "@1", "@2", etc. contained in the "dims" query parameter.

Each line in the file can also contain an options object: Start the line with "={...}" as the first parameter. Values of the object within {...} are:

desc - a description that is returned to the caller as the first value for each sub-array.

mult - a multiplier applied to numeric values

dec - number of digits to right of decimal

form - format specifier, contains one or more of these characters:

"$" to format as currency

"%" to format as percentage

"," to use 1000's separators

"(" to format negative numbers in parenthesis

Note: If "dec" or "form" are specified, converted string values are returned instead of numbers.

Here is a sample of a definition file:

={"desc":"Rooms Revenue","mult":-1,"dec":0,"form":"$,"};"Hotel Company";"Rooms";"All Revenue Accounts"

={"desc":"Food Revenue","mult":-1,"dec":2,"form":","};"Beach Club";"Food & Beverage";"Total Food Revenue"

In this case, dims would have @1, @2, @3 corresponding to the 3 element values in the file.

**Other usage notes**

1. A current restriction is that dimension and element names are case sensitive, so that exact proper case must be used.

**Service configuration**

The **config.json** file contains service parameters. Most of these can also be specified as environment variables for the process running the service. Parameters are:

"palo\_server": The OLAP server IP address/

"palo\_userid": User name for login to the server

"palo\_passwd": Password. Either plain text or encrypted as "0x……"

"max\_sessions\_per\_server": Maximum connections to the server

"save\_sessions\_per\_server": Number of idle connections to save

"log\_file" : Write log to a daily file in the “log” folder.

"log\_format": Format string for morgan logger

"palo\_allowdatabases": A list of permitted databases. Only databases in this list can be accessed.

"palo\_allowsetdata": true to allow data updates

Note: An improved security model is needed to update or replace these last 2 settings and possibly the OLAP login also.