Public Query Interface

# Document history

2015-03-02 Dave Snelling – Initial draft.

2015-05-29 Dave Snelling – Column names fixed to match atom field names.

# Introduction

This document describes the minimum synchronous query interface that must be provided by a Data Engine. Individual implementations of a Data Engine may provide further capabilities.

# Query Interface

The query interface has one method POST. The content type of the request and response is "application/json". The body of the request contains the query. The response to a successful query is a list of JSON Atoms that are the results of the query or the result of an aggregation.

# Query Language

Queries are JSON documents. The query structure is described below:

* ServiceProviderID: ID of the service provider (required)
* CoelitionID: user ID (required)
* Timewindow: (optional)
	+ StartTime: start time (seconds since 1/1/1970) (optional)
	+ EndTime: end time (seconds since 1/1/1970) (optional)
	+ BlockBy: if present the number of seconds in each block returned (optional). If absent all Atoms in the time window are returned as a single block or used in the aggregation computation.
* Query: (required)
	+ Filter:
		- ColName: column name
		- Comparator: one of "=", ">", ">=", "<", "<=", "!="
		- Value: comparison value
	+ AND (list of length > 0) (optional)
		- Filter, AND, OR
	+ OR (list of length > 0) (optional)
		- Filter, AND, OR
	+ NOT (optional)
		- Filter, AND, OR
	+ Aggregate (optional)
		- Columns (list)
			* ColName: column name, see below
			* Aggregator: aggregator function, one of AVG, SUM, COUNT, MIN, MAX, STDDEV
		- GroupBy (list) (optional)
			* ColName: column name
	+ Project (optional)
		- Include (list)
			* ColName: column name
		- Exclude (list)
			* ColName: column name

# Column names in Query

|  |  |
| --- | --- |
| **Name** | **Data type** |
| HEADER\_VERSION | short |
| WHEN\_TIMEZONE | int |
| WHEN\_ACCURACY | int |
| WHEN\_DURATION | int |
| WHAT\_CLUSTER | short |
| WHAT\_CLASS | short |
| WHAT\_SUBCLASS | short |
| WHAT\_ELEMENT | short  |
| HOW\_HOW | int |
| HOW\_CERTAINTY | int |
| HOW\_RELIABILITY | int |
| CONTEXT\_SOCIAL | int |
| CONTEXT\_WEATHER | int |
| CONTEXT\_CONTEXTTAG | int |
| CONTEXT\_CONTEXTVALUE | int |
| WHERE\_EXACTNESS | int |
| WHERE\_LATITUDE | double |
| WHERE\_LONGITUDE | double |
| WHERE\_MCC | int |
| WHERE\_MNC | int |
| WHERE\_LCA | int |
| WHERE\_CID | int |
| WHERE\_PLACE | int |
| WHERE\_POSTCODE | String |
| EXTENSION\_INTTAG | int |
| EXTENSION\_INTVALUE | int |
| EXTENSION\_FLTTAG | int |
| EXTENSION\_FLTVALUE | double |
| EXTENSION\_STRTAG | int |
| EXTENSION\_STRVALUE | string |

# Query Result

The query result is a list of JSON objects that match the query.

For a simple filter the result is a list of Atoms. If a projection is specified only requested columns of the matching Atoms are included.

For aggregates, the result objects contain a list of aggregated columns, described by column name and aggregator (as specified in the query), with the result of the aggregate function. If a grouping is specified the object contains a list of column names and their groups for each aggregation.

When BlockBy is absent, all results are returned as the only element in the Blocks list.

* Blocks:(list)
	+ Aggregate: (list)
		- ColName: column name
		- Aggregator: aggregate function
		- Value: aggregate function value
	+ Group: (list)
		- ColName: grouping column
		- Value: group

# Minimum Implementation

Any implementation MUST accept queries in the form described above, however only a minimum functionality must be supported.

A Data Engine MUST return raw atoms within a time window.

A Data Engine MUST return the number of atoms held in a time window for a CoelitionID.

# Example Queries

The following is one of the two minimum queries that a Data Engine implementation must support.

{

"ServiceProviderID" : 17,
 "CoelitionID" : "ed58fc40-a866-11e4-bcd8-0800200c9a66",
 "Timewindow" : {

   "StartTime" : 1415145600,

  "EndTime" : 1415232000

}

}

The result of this query is a list of all Atoms with a start time within the time window.

The following is the other query that a Data Engine implementation must support.

{

"ServiceProviderID" : 17,
 "CoelitionID" : "ed58fc40-a866-11e4-bcd8-0800200c9a66",
 "Timewindow" : {

   "StartTime" : 1415145600,

  "EndTime" : 1415232000

},

  "Aggregate" : {

 "ColName" : "WHAT\_CLUSTER",

 "Aggregator" : "COUNT"}

  }

}

The result of this query is the number of Atoms with a start time within the time window.