New HTTP API Design

Motivation

The purpose of this document is to outline the design of refactored HTTP (aka REST) API for AsterixDB.

The reason for HTTP API refactoring is threefold:

- Currently we have various REST endpoints (/ddl, /query, /update, /aql, /query/service), which do similar things (parsing parameters out of the request parameters, assembling the response) but each endpoint does it in a slightly different manner. All of that should be refactored to use the common base endpoint. At the same time old endpoints could be kept for backwards-compatibility if that it really needed.
- Current Web interface does the server-side HTML generation and does not use REST calls whatsoever. It would be better to eat our own dog food here and switch to assembling the result on the client side using JavaScript. Especially since we have a potential GSoC project proposal to build such JS-based interface.
- As we are moving towards having more query languages under AsterixDB umbrella (AQL, SQL++, XQuery/JSONiq as a part of VXQuery) it would be nice to design a generic language-agnostic REST API, which later could be reused by VXQuery since it's also lacking proper API as of now.

Tools and existing APIs

The design is inspired by N1QL REST API (http://developer.couchbase.com/documentation/server/4.5/n1ql/n1ql-rest-api/executen1ql.html) since it is an example of well thought API in the similar system. I believe we don't need to be 100% compatible, although it would be nice to be able to reuse the same clients.

We might also consider using Swagger (http://swagger.io/) to describe the API. This library will allow users to seamlessly generate client SDKs in their favorite language, which is especially usefully since we don't provide drivers for any clients. Here is the complete set of features which Swagger will allow us to do:

- Describe API in developer-friendly way by creating yaml description.
- Validate correctness of the server-side implementation.
- Generate client-side SDK for various languages.
- Generate documentation for API.

REST Endpoints

Proposed API consists of 3 endpoints: Query, Status and Result. The latter two are needed only in a case of asynchronous data delivery, while the former is the main endpoint and serves as an entry point for all client requests.

Query Endpoint (/query/service)

The following compares various parameters & HTTP headers using in N1QL API and in current AsterixDB API and proposes parameters to be used in new API.

HTTP Request(POST) parameters:

<table>
<thead>
<tr>
<th>N1QL API Parameter</th>
<th>Value</th>
<th>Old Asterix API Parameter</th>
<th>Proposed API Parameter</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>statement</td>
<td>string</td>
<td>query</td>
<td>statement</td>
<td>A semicolon separated sequence of AQL/SQL++ statements (DDL, update /load statement, FLOWR query), which should be executed. The result of the last statement is returned.</td>
</tr>
<tr>
<td>format</td>
<td>enum</td>
<td>Accept HTTP header</td>
<td>format</td>
<td>[Optional] Desired format for the query results. Possible values are ADM, JSON, CSV. (default: ADM)</td>
</tr>
<tr>
<td>Proposed API Parameter</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>signature</td>
<td>[Optional] Defines whether to include a header for the results schema in the response. (default: false) In case of CSV result format header is included right into the result.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>include-results</td>
<td>[Optional] Defines whether to include results right into the response, or return a handle to retrieve them. (default: true)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mode</td>
<td>[Optional] Result delivery mode. Possible values are immediate, deferred, async. (default: immediate)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lossless</td>
<td>[Optional] Defines whether to use lossless-JSON output for JSON-encoded output or keep clean-JSON instead. (default: false)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wrapper-array</td>
<td>[Optional] Defines whether to wrap ADM-encoded output into array-brackets. (default: false) Used only when format=ADM and include-results=false.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>expr-tree</td>
<td>[Optional] Defines whether to include an query expression AST into the result (default: false)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rewritten-expr-tree</td>
<td>[Optional] Defines whether to include a rewritten query expression AST into the result (default: false)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>logical-plan</td>
<td>[Optional] Defines whether to include a logical plan into the result (default: false)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>optimized-plan</td>
<td>[Optional] Defines whether to include an optimized logical plan into the result (default: false)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hyracks-job</td>
<td>[Optional] Defines whether to include a Hyracks job into the result (default: false)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>execute-query</td>
<td>[Optional] Defines whether to execute a statement (default: true)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**HTTP Response format:**

Type of the response - JSON document (even if format in request was CSV it will be serialized as one of JSON attributes)

<table>
<thead>
<tr>
<th>N1QL API</th>
<th>Value</th>
<th>Old Asterix API</th>
<th>Proposed API Parameter</th>
<th>Comment</th>
</tr>
</thead>
</table>
| results  | JSONList| HTTP response body | results               | One of 2 possible values
1) A list of all results returned by the query (mode=synchronous and include-results=true).
2) No value (statement is DDL/update/load) |
<table>
<thead>
<tr>
<th><strong>type</strong></th>
<th><strong>String</strong></th>
<th><strong>MIME type of result</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>URI</td>
<td>handle</td>
</tr>
<tr>
<td>signature</td>
<td>JSONObject</td>
<td>signature</td>
</tr>
<tr>
<td>status</td>
<td>string</td>
<td>status</td>
</tr>
<tr>
<td>errors</td>
<td>JSONList</td>
<td>errors</td>
</tr>
<tr>
<td>error.code</td>
<td>int</td>
<td>error.code</td>
</tr>
<tr>
<td>error.msg</td>
<td>string</td>
<td>summary</td>
</tr>
<tr>
<td>error.name</td>
<td>string</td>
<td>-</td>
</tr>
<tr>
<td>error.sev</td>
<td>int</td>
<td>-</td>
</tr>
<tr>
<td>error.temp</td>
<td>boolean</td>
<td>-</td>
</tr>
<tr>
<td>string</td>
<td>stacktrace</td>
<td>error.stacktrace</td>
</tr>
<tr>
<td>warnings</td>
<td>JSONList</td>
<td>warnings</td>
</tr>
<tr>
<td>metrics</td>
<td>JSONObject</td>
<td>metrics</td>
</tr>
<tr>
<td>metrics.executionTime</td>
<td>string</td>
<td>metrics.executionTime</td>
</tr>
<tr>
<td>metrics.resultCount</td>
<td>unsigned int</td>
<td>metrics.resultCount</td>
</tr>
<tr>
<td>-</td>
<td>JSONObject</td>
<td>plans</td>
</tr>
<tr>
<td>-</td>
<td>JSONObject/string</td>
<td>plans.exprTree</td>
</tr>
<tr>
<td>-</td>
<td>JSONObject/string</td>
<td>plans.rewrittenExprTree</td>
</tr>
<tr>
<td>-</td>
<td>JSONObject/string</td>
<td>plans.logicalPlan</td>
</tr>
<tr>
<td>-</td>
<td>JSONObject/string</td>
<td>plans.optimizedPlan</td>
</tr>
<tr>
<td>-</td>
<td>JSONObject/string</td>
<td>plans.hyracksJob</td>
</tr>
</tbody>
</table>
Examples:

1. DDL request

   Request
   
   curl -v http://localhost:19002/query/service -X POST \
   -d "statement=create dataverse test;"

   Response
   
   < HTTP/1.1 200 OK
   Content-Type: application/json
   {
     "status": "success"
     "metrics": {
       "executionTime": "1ms"
     }
   }

2. Query which is not executed, but returns logical plan

   Request
   
   curl -v http://localhost:19002/query/service -X POST \
   -d "statement=for $x in dataset testDS return $x & lossless=true & 
   logical-plan=true & execute-query=false"

   Response
   
   < HTTP/1.1 200 OK
   Content-Type: application/json
   {
     "status": "success"
     "metrics": {
       "executionTime": "5ms"
     },
     "plans": {
       "logical_plan": [
         {"operator": "distribute_result", "args": [{"exp": "var_ref",
         "var": "$\$0"}]},
         {"operator": "datascan", "output_vars": ["$$0", "$\$1"]}
       ]
     }
   }

3. Query which synchronously returns CSV (with header) result inside JSON response
3. Request

-d "statement=for $x in dataset Tweets return $x & signature=true"

4. Response

< HTTP/1.1 200 OK
Content-Type: application/json
{
  "status": "success",
  "results": "'id','screen_name','message_text'\n1','BarackObama','Four more years'"
  "metrics": {
    "executionTime": "10ms",
    "resultCount": 1
  }
}  

4. Query which returns optimizer error

Request

curl -v http://localhost:19002/query/service -X POST \
-d "statement=create dataset Tweets(TweetType) primary key facebook_id"

Response

< HTTP/1.1 400 Bad Request
Content-Type: application/json
{
  "status": "error",
  "error": {
    "code": 1,
    "msg": "Field 'facebook_id' cannot be found in datatype 'TweetType'"
  }
  "metrics": {
    "executionTime": "1ms"
  }
}

5. Query which returns runtime error

6. Query which runs synchronously, however does not include them in the response, but provides a handle to retrieve them. Request also specifies to include ADM type (signature).
7. Query which returns lossless-JSON results asynchronously and does not include them in the response, but provides a handle to retrieve them.

Request

```
-d "statement=for $x in dataset Tweets return $x & lossless=true & mode=asynchronous & include-results=false"
```

Response

```
< HTTP/1.1 200 OK
Content-Type: application/json
{
   "status": "success",
   "handle": "http://localhost:19002/query/service/result/27-2",
   "signature": {
      "id": "int64",
      "screen_name": "string",
      "message_text": "string"
   },
   "metrics": {
      "executionTime": "10ms",
      "resultCount": 2
   }
}
```

Status Endpoint (/query/service/status)

This endpoint is supposed to be used only in the case when results are delivered asynchronously. The endpoint purpose is solely to inform about status of submitted query, and possibly include results of its execution.

HTTP Request(GET) format:

```
http://localhost:19002/query/service/status/ID
```

Where ID is an identifier generated and returned by the /query/service endpoint.
HTTP Response parameters:

The response is a small subset of response of the /query/service endpoint

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>enum</td>
<td>The status of the request; possible values are: success (query is completed), running (query is still running), fatal (execution error).</td>
</tr>
<tr>
<td>results</td>
<td>JSONObject</td>
<td>The URI to /query/service/result endpoint, if the query was completed (status=succcess and include-results=true).</td>
</tr>
<tr>
<td>handle</td>
<td>URI</td>
<td>The URI to /query/service/result endpoint, if the query was completed (status=succcess and include-results=false).</td>
</tr>
<tr>
<td>error</td>
<td>JSONObject</td>
<td>An object containing details of the error.</td>
</tr>
<tr>
<td>error.code</td>
<td>int</td>
<td>A code that identifies the error.</td>
</tr>
<tr>
<td>error.msg</td>
<td>string</td>
<td>A message describing the error in detail.</td>
</tr>
<tr>
<td>error.stacktrace</td>
<td>string</td>
<td>A stack trace of the error.</td>
</tr>
<tr>
<td>metrics</td>
<td>JSONObject</td>
<td>An object containing details of the execution metrics (only when status=succcess).</td>
</tr>
<tr>
<td>metrics.executionTime</td>
<td>string</td>
<td>The time it took to execute the request</td>
</tr>
<tr>
<td>metrics.resultCount</td>
<td>int</td>
<td>The total number of objects in the results</td>
</tr>
</tbody>
</table>

Examples:

1. Query which returns runtime error

   **Request**
   ```bash
   ```

   **Response**
   ```json
   < HTTP/1.1 500 Internal Server Error
   Content-Type: application/json
   {
   "status": "fatal",
   "error": {
   "code": 99,
   "msg": "Something happen during query execution",
   "stacktrace": "java.lang.RuntimeException: \n\r at java.lang.Thread.run(Thread.java:745)"
   }
   }
   ```

2. Query which is still executing
3. Query which successfully completes and return URI to its results. Refer to Query endpoint Example 7 to see the original request.

```
Request


Response

`< HTTP/1.1 200 OK
Content-Type: application/json
{
  "status": "success",
  "handle": "http://localhost:19002/query/service/result/27-2"
  "metrics": {
    "executionTime": 100ms,
    "resultCount": 10
  }
}`
```

Result Endpoint (/query/service/result)

This endpoint is used to retrieve results of asynchronous query execution or synchronous results, when the client opted out from having results included into /query request (include-results=false).

Note that the Content-type of the result is set up appropriately to Accept in the initial request.

**HTTP Request(GET) format:**

`http://localhost:19002/query/service/result/ID`

Where ID is an identifier generated and returned by the /query/service endpoint (include-results=false) or by /query/service/status endpoint when the asynchronous result is computed.

**HTTP Response format:**

The response includes only the results with appropriate Content-Type header.

**Examples:**
1. Synchronous query results. Refer to Query endpoint Example 6 to see the original request.

Request

curl -v http://localhost:19002/query/service/result/27-2 -X GET

Response

HTTP/1.1 200 OK
Content-Type: application/x-adm
{
   "id": 1,
   "screen_name": "BarackObama",
   "message_text": "Four more years"
}
{
   "id": 2,
   "screen_name": "ElonMusk",
   "message_text": "I Would Like to Die on Mars, Just Not on Impact"
}

2. Asynchronous query results. Refer to Query endpoint Example 7 & Status endpoint Example 3 to see the original requests.

Request

curl -v http://localhost:19002/query/service/result/27-2 -X GET

Response

HTTP/1.1 200 OK
Content-Type: application/json
[
   {
      "id": 1,
      "screen_name": "BarackObama",
      "message_text": "Four more years"
   },
   {
      "id": 2,
      "screen_name": "ElonMusk",
      "message_text": "I Would Like to Die on Mars, Just Not on Impact"
   }
]