Server Product Specification

Table of Contents

1Schema Management 2

2Extension Management 3

3API Reference 4

3.1 General 4

3.2 Methods 4

3.2.1 /event 4

3.3 /session 5

3.3.1 GET /session/:id 5

3.3.2 PUT /session/:id 7

4Future 8

# Schema Management

## General

Schema files are read once during server startup from a file located inside the schema deploy directory. If an error is encountered, the server must be restarted after it is corrected.

Only a single schema file per server is supported at this time.

The schema is given the name of the schema file, with the extension (everything, including and following the last dot) removed. Schema name is not case sensitive.

If the server has encountered a fatal error, all subsequent connection request will receive HTTP status 503 “Service Unavailable.”

## Schema deploy directory location

Schema files are located in the OS directory specified by, in the order of significance

* -Dvariant.schemas.dir system property
* variant.data.dir configuration property
* (/schemas classpath directory — future improvement?)

Value is treated the same as Java’s File(String), i.e. if starts with slash is understood as absolute path, otherwise as relative to the applicatioin’s running directory.

# Extension Management

# API Reference

## General

* Property names are case insensitive, i.e. createDate is the same as CreateDate.
* Sessions are stored on the server as serialized JSON strings and are deserialized lazily, if server needs them.

## Methods

Notation:

"name": <Number?=NOW>

* **"name":** name to the left of colon is the literal property name. (
* **<Number?=NOW>** information inside the angle brackets is the data type (String/Number/Boolean/Array/Object), optionally followed by the question mark, indicating that this field is optional, optionally followed by the equal sign and the default value. If no question mark, this field is required. If question mark, but no equal sign, the field is optional with no default.
* Note, that the type specification

### /event

#### POST /event

Trigger a user-defined event.

|  |  |  |
| --- | --- | --- |
| Response Headers | | |
| Content-Type | text/plain; charset=utf-8 | |
| Request Body | | |
| {  "sid": <String>,  "name": <String>,  "value": <String?>,  "ts": <Number?=NOW>,  "params": <Array?> [  {  "name": <String>,  "value": <String?>  },  ...  ],  } | | |
| session-id | | Current variant session ID. |
| event-name | | The name of the event. Any string. |
| event-value | | Value of the event. Any string. |
| event-create-date | | Event creation date, as Epoch time. Optional. If not given, defaults to now. |
| event-parameters | | A map of event parameters. Optional. |
| Response Headers | | |
|  | | |
| Response body | | |
| None. | | |
| Response Codes | | |
| 200 OK | | Event triggered. |
| 400 BAD\_REQUEST | | Invalid request body. One of a number of user errors was detected in the request body, e.g. a missing required parameter. An additional message will be provided in the HTTP Status header. |
| 403 FORBIDDEN | | Invalid session ID. The session ID provided is not known. |

## /session

|  |  |
| --- | --- |
| GET /session/:id | |
| Get session by ID.   |  |  |  | | --- | --- | --- | | Headers | | | | Content-Type | text/plain; charset=utf-8 | | | Request Body | | | | None | | | | Response body | | | | {  "sid": session-id::String,  "ts": timestamp::Number,  "schid": schema-id::String,  "req": request-def::Object ?,  "states": [  {  "state": state-name::String,  "count": visit-count::Number,  },  ...  ],  "tests": [test-name::String,...]  "disqualTests": [test-name::String,...]  }  request-def ::=  {  "state": state-name::String,  "status": status::String,  "comm": is-commited::Boolean,  "params": [  {  "key": param-name::String,  "val": param-value::String  },  ...  ],  "exps": experience-list::List[String]  } | | | | session-id | | Current variant session ID. | | timestamp | | Session creation timestamp, as the Unix Epoch time, i.e. the number of milliseconds since January 1, 1970, 00:00:00 GMT | | schema-id | | Variant schema ID in effect when this session was created. | | state-name | | The name of a visited state. | | visit-count | | Number of times this state has been visited by this session. | | test-name | | The name of a traversed test | | is-qualified | | Is this session qualified for this test. | |  | | | | Response Headers | | | |  | | | | Response Codes | | | | 200 OK | | Session found and returned. | | 404 NOT\_FOUND | | No session with that ID on server. |  PUT /session/:id | |
| Save or replace user session by session ID. Idempotent. Body is not parsed but saved in the session store under the ID. Only parsed if required instantiation on Server. |  |

|  |  |  |
| --- | --- | --- |
| Headers | | |
| Content-Type | text/plain; charset=utf-8 | |
| Request Body | | |
| See response body of GET /session/:id | | |
| Response body | | |
| None. | | |
| Response Codes | | |
| 200 OK | | Session created or replaced. |
| 400 BAD\_REQUEST | | Invalid request body. One of a number of user errors was detected in the request body, e.g. a missing required parameter. An additional message will be provided in the HTTP Status header. |

# Future