Variant Product Roadmap

As Of Release 0.7

# Core

## Schema Parser

|  |  |
| --- | --- |
| 1.1.a | Better Grammar Than JSON |
| Problems with JSON:   * No variables * Too cryptic for a human to read/write. * Current parsing unit is the entire schema. Need to create/delete/update tests and states independently. | |

|  |  |
| --- | --- |
| 1.1.b | Custom Schema Parser |
| Need to retain annotated parse tree until semantic phase so we can provide line/column in all errors. | |

# Server

## Configuration

Nothing.

## Schema Management

|  |  |
| --- | --- |
| 2.2.a | Multiple Schemas In Directory |
| Support multiple schemas in the schemas directory.   * At startup, schemas that parsed will be deployed. | |

|  |  |
| --- | --- |
| 2.2.a | Multiple Schemas In a File |
| A schema file format:  [  {…},  {…}  ]  Each schema is an independent deployment unit, i.e. those that succeeded will be deployed. | |

|  |  |
| --- | --- |
| 2.2.b | Schema Reload |
| Server must detect a new schema file and (re)deploy the schema(s). | |

## API

Nothing.

## Authentication

* From trusted server via a client
* From untrusted Web browser via JavaScript client
* From untrusted mobile app via Java client

### Trusted Server

Connections from Java or server side JavaScript clients to Variant server running behind the client’s firewall. On-premises or in the cloud. . No need for any client side participation. Server can either trust or whitelist known client IP addresses, like HBA in Postgres.

### Untrusted Web Browser

Untrusted in the sense that whatever secret we might require it to send will have to be stored there. Can be recovered by a sophisticated intruder and used for abusing the API, e.g. stealing bandwidth from the lawful key owner, or acting on his behalf.

Disallow new session creation from remote JavaScript. Instead, pass the session ID token from the trusted server to variant.js. Session ID becomes a required parameter at API creation time. All other calls are okay from remote JS.

Local JavaScript (e.g. server-side Node.js)  still needs to be able to create session.

### Untrusted Mobile Client

Similar to remote JS, in that if the secret is stored in the application code, a malitious user can figure it out and abuse the API. Typically, O-Auth2, but we can’t use that because our user is not the end-user, but the developer of the app.

Same solution:

## Schema Management

### Hot-Deployment

Schema files are located inside the schemata directory. They are hot-deployed, i.e. whenever a schema file changes, or a new one is added, the running server detects the change and attempts to parse and (re)deploy.

If a fatal parsing error is encountered the currently deployed schema by that name remains deployed. Otherwise, the new schema is deployed and replaces the previously deployed schema by that name.

Until then, all subsequent connection request will receive HTTP status 503 “Service Unavailable.”

Only a single schema file per server, e.g. per deploy directory, is supported at this time.

The schema file must start with the meta section:

{

"meta": {

"name": schema-name::String,

"comment": schema-comment::String?

},

“states”: {...},

“tests”: {...)

}

### Schemata directory location

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

* -Dvariant.schemata.dir system property
* variant.schemata.dir configuration property
* (/schemata 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 running server’s current directory.

Cannot be changed in the running server.

## Server Admin Console

<http://path.to/variant/admin>

Displays: version, uptime, list of currently deployed schemas, deployment directory, number of active connections to the schema and sessions on each of the connections.

Allows to change (?)

Undeploy a deployed schema. (What happens to the connections and session?)

Redeploy currently undeployed schema.

# Serialization

* Replace json with a more efficient protocol, e.g. [Protocol Buffer](https://github.com/google/protobuf).

# Adapters

## **Web-Java**