NAME

ovsdb − Open vSwitch Database (File Formats)

DESCRIPTION

OVSDB, the Open vSwitch Database, is a database system whose network protocol is specified by RFC 7047. The RFC does not specify an on−disk storage format. The OVSDB implementation in Open vSwitch implements two storage formats: one for standalone (and active−backup) databases, and the other for clustered databases. This manpage documents both of these formats.

Most users do not need to be concerned with this specification. Instead, to manipulate OVSDB files, refer to *ovsdb−tool(1)*. For an introduction to OVSDB as a whole, read *ovsdb(7)*.

OVSDB files explicitly record changes that are implied by the database schema. For example, the OVSDB "garbage collection" feature means that when a client removes the last reference to a garbage−collected row, the database server automatically removes that row. The database file explicitly records the deletion of the garbage−collected row, so that the reader does not need to infer it.

OVSDB files do not include the values of ephemeral columns.

Standalone and clustered database files share the common structure described here. They are text files encoded in UTF−8 with LF (U+000A) line ends, organized as append−only series of records. Each record consists of 2 lines of text.

The first line in each record has the format **OVSDB <magic> <length> <hash>**, where <magic> is **JSON** for standalone databases or **CLUSTER** for clustered databases, <length> is a positive decimal integer, and <hash> is a SHA−1 checksum expressed as 40 hexadecimal digits. Words in the first line must be separated by exactly one space.

The second line must be exactly *length* bytes long (including the LF) and its SHA−1 checksum (including the LF) must match *hash* exactly. The line's contents must be a valid JSON object as specified by RFC 4627. Strings in the JSON object must be valid UTF−8. To ensure that the second line is exactly one line of text, the OVSDB implementation expresses any LF characters within a JSON string as **\n**. For the same reason, and to save space, the OVSDB implementation does not "pretty print" the JSON object with spaces and LFs. (The OVSDB implementation tolerates LFs when reading an OVSDB database file, as long as *length* and *hash* are correct.)

JSON Notation

We use notation from RFC 7047 here to describe the JSON data in records. In addition to the notation defined there, we add the following:

<raw−uuid>

A 36−character JSON string that contains a UUID in the format described by RFC 4122, e.g. **"550e8400−e29b−41d4−a716−446655440000"**

Standalone Format

The first record in a standalone database contains the JSON schema for the database, as specified in RFC 7047. Only this record is mandatory (a standalone file that contains only a schema represents an empty database).

The second and subsequent records in a standalone database are transaction records. Each record may have the following optional special members, which do not have any semantics but are often useful to administrators looking through a database log with **ovsdb−tool show−log**:

"_date": <integer>

The time at which the transaction was committed, as an integer number of milliseconds since the Unix epoch. Early versions of OVSDB counted seconds instead of milliseconds; these can be detected by noticing that their values are less than 2**32.

OVSDB always writes a **_date** member.

"_comment": <string>
A JSON string that specifies the comment provided in a transaction **comment** operation. If a transaction has multiple **comment** operations, OVSDB concatenates them into a single **_comment** member, separated by a new−line.

OVSDB only writes a **_comment** member if it would be a nonempty string.

Each of these records also has one or more additional members, each of which maps from the name of a database table to a <table−txn>:

<table−txn>
A JSON object that describes the effects of a transaction on a database table. Its names are <raw−uuid>s for rows in the table and its values are <row−txn>s.

<row−txn>
Either **null**, which indicates that the transaction deleted this row, or a JSON object that describes how the transaction inserted or modified the row, whose names are the names of columns and whose values are <value>s that give the column's new value.

For new rows, the OVSDB implementation omits columns whose values have the default values for their types defined in RFC 7047 section 5.2.1; for modified rows, the OVSDB implementation omits columns whose values are unchanged.

Clustered Format
The clustered format has the following additional notation:

<uint64>
A JSON integer that represents a 64−bit unsigned integer. The OVS JSON implementation only supports integers in the range −2**63 through 2**63−1, so 64−bit unsigned integer values from 2**63 through 2**64−1 are expressed as negative numbers.

<address>
A JSON string that represents a network address to support clustering, in the **<protocol>:<ip>:<port>** syntax described in **ovsdb−tool(1)**.

<servers>
A JSON object whose names are <raw−uuid>s that identify servers and whose values are <address>es that specify those servers' addresses.

<cluster−txn>
A JSON array with two elements:

1. The first element is either a <database−schema> or **null**. A <database−schema> element is always present in the first record of a clustered database to indicate the database's initial schema. If it is not **null** in a later record, it indicates a change of schema for the database.
2. The second element is either a transaction record in the format described under **Standalone Format** above, or **null**.

When a schema is present, the transaction record is relative to an empty database. That is, a schema change effectively resets the database to empty and the transaction record represents the full database contents. This allows readers to be ignorant of the full semantics of schema change.

The first record in a clustered database contains the following members, all of which are required, except **prev_election_timer**:

"server_id": <raw−uuid>
The server's own UUID, which must be unique within the cluster.

"local_address": <address>
The address on which the server listens for connections from other servers in the cluster.

"name": <id>
The database schema name. It is only important when a server is in the process of joining a cluster: a server will only join a cluster if the name matches. (If the database schema name were unique, then we would not also need a cluster ID.)

"cluster_id": <raw−uuid>
The cluster's UUID. The all−zeros UUID is not a valid cluster ID.

"prev_term": <uint64> and "prev_index": <uint64>
The Raft term and index just before the beginning of the log.

"prev_servers": <servers>
The set of one or more servers in the cluster at index "prev_index" and term "prev_term". It might not include this server, if it was not the initial server in the cluster.

"prev_election_timer": <uint64>
The election base time before the beginning of the log. If not exist, the default value 1000 ms is used as if it exists this record.

"prev_data": <json−value> and "prev_eid": <raw−uuid>
A snapshot of the data in the database at index "prev_index" and term "prev_term", and the entry ID for that data. The snapshot must contain a schema.

The second and subsequent records, if present, in a clustered database represent changes to the database, to the cluster state, or both. There are several types of these records. The most important types of records directly represent persistent state described in the Raft specification:

Entry A Raft log entry.

Term The start of a new term.

Vote The server's vote for a leader in the current term.

The following additional types of records aid debugging and troubleshooting, but they do not affect correctness.

Leader Identifies a newly elected leader for the current term.

Commit Index
An update to the server's **commit_index**.

Note A human−readable description of some event.

The table below identifies the members that each type of record contains. "yes" indicates that a member is required, "?" that it is optional, blank that it is forbidden, and [1] that **data** and **eid** must be either both present or both absent.

member	Entry	Term	Vote	Leader	Commit Index	Note
comment	?	?	?	?	?	?
term	yes	yes	yes	yes		
index	yes					
servers	?					
election_timer	?					
data	[1]					
eid	[1]					
vote			yes			

leader				yes		
commit_index					yes	
note						yes

The members are:

"comment": <string>
A human−readable string giving an administrator more information about the reason a record was emitted.

"term": <uint64>
The term in which the activity occurred.

"index": <uint64>
The index of a log entry.

"servers": <servers>
Server configuration in a log entry.

"election_timer": <uint64>
Leader election timeout base value in a log entry.

"data": <json−value>
The data in a log entry.

"eid": <raw−uuid>
Entry ID in a log entry.

"vote": <raw−uuid>
The server ID for which this server voted.

"leader": <raw−uuid>
The server ID of the server. Emitted by both leaders and followers when a leader is elected.

"commit_index": <uint64>
Updated **commit_index** value.

"note": <string>
One of a few special strings indicating important events. The currently defined strings are:

"transfer leadership"
This server transferred leadership to a different server (with details included in **comment**).

"left" This server finished leaving the cluster. (This lets subsequent readers know that the server is not part of the cluster and should not attempt to connect to it.)

Joining a Cluster

In addition to general format for a clustered database, there is also a special case for a database file created by **ovsdb−tool join−cluster**. Such a file contains exactly one record, which conveys the information passed to the **join−cluster** command. It has the following members:

"server_id": <raw−uuid> and "local_address": <address> and "name": <id>
These have the same semantics described above in the general description of the format.

"cluster_id": <raw−uuid>
This is provided only if the user gave the **−−cid** option to **join−cluster**. It has the same semantics described above.

"remote_addresses"; [<address>*]
One or more remote servers to contact for joining the cluster.

When the server successfully joins the cluster, the database file is replaced by one described in *Clustered Format*.

AUTHOR

The Open vSwitch Development Community

COPYRIGHT