OVSDB Library Developer Guide

Overview

The OVSDB library manages the Netty connections to network nodes and handles bidirectional JSON-RPC messages. It not only provides OVSDB protocol functionality to OpenDaylight OVSDB plugin but also can be used as standalone JAVA library for OVSDB protocol.

The main responsibilities of OVSDB library includes:

- Manage connections to peers
- Marshal and unmarshal JSON Strings to JSON objects.
- Marshal and unmarshal JSON Strings from and to the Network Element.

Connection Service

OVSDB library provides connection management through OvsdbConnection interface. OvsdbConnection interface provides OVSDB connection management APIs which includes both active and passive connections. From the library perspective, active OVSDB connections are initiated from the controller to OVS nodes while passive OVSDB connection are initiated from OVS nodes to controller. In the active connection scenario, application needs to provide IP address and listening port on OVS nodes to library management API. On the other hand, the library management API only requires the info of controller listening port in passive connection scenario.

For passive connection scenario, the library also provides connection event listener through OvsdbConnectionListener interface. The listener interface has connected() and disconnected() methods to notify application when a new passive connection is established or an existing connection is terminated.

SSL Connection

In additional to regular TCP connection, the OvsdbConnection interface also provides connection management API for SSL connection. To start OVSDB connection with SSL, application will need to provide an Java SSLContext object to the management API. There could be different ways to create JAVA SSLContext, but in most case a Java KeyStore with certificate and private key provided by the application is required. Detail steps about how to create Java SSLContext is out of the scope of this document and can be found in Java doc JAVA Class SSlContext.

In active connection scenario, the library uses given SSLContext to create Java SSLEngine and configure the SSL engine with client mode for SSL handshaking. Normally clients are not required to authenticate themselves.

In the passive connection scenario, the library uses given SSLContext to create Java SSLEngine which will operate on server mode for SSL handshaking. For security reason, SSLv3 protocol and some cipher suites are disabled. Currently the OVSDB server only supports TLS_RSA_WITH_AES_128_CBC_SHA cipher suite and following protocols: SSLv2Hello, TLSv1, TLSv1.1, TLSv1.2.

The SSL engine is also configured to operate on two-way authentication mode for passive connection scenario, i.e, the OVSDB server (controller) will authenticate clients (OVS nodes) and clients (OVS nodes)l are also required to authenticate the server (controller). In two-way authentication mode, application should keep a trust manager to store certificates of trusted clients and initialize Java SSLContext with this trust manager. Thus during SSL handshaking process the OVSDB server (controller) can use the trust manager to verify clients and only accept connection request from trusted clients. On the other hand, user should also configure OVS nodes to authenticate the controller. OpenVswitch already supports this functionality in ovsdb-server command with option `--ca-cert=cacert.pem` and `--bootstrap-ca-cert=cacert.pem`. On OVS node, user can use option `--ca-cert=cacert.pem` to specify controller certificate directly and the node will only allow connection to controller with specified certificate. If the OVS node runs ovsdb-server with option `--bootstrap-ca-cert=cacert.pem`, it will authenticate the controller with the specified certificate cacert.pem. If the certificate file doesn't exist, it will attempt to obtain a certificate from the peer (controller) on its first SSL connection and save it to the named PEM file `cacert.pem`. Here is an example of ovsdb-server with `--bootstrap-ca-cert=cacert.pem` option:

```
ovsdb-server --pidfile --detach --log-file --remote punix:/var/run/openvswitch/db.sock --remote=db:hardware_vtep,
Global,managers --private-key=/etc/openvswitch/ovsclient-privkey.pem -- certificate=/etc/openvswitch/ovsclient-
cert.pem --bootstrap-ca-cert=/etc/openvswitch/vswitchd.cacert
```

OVSDB protocol transactions

The OVSDB protocol defines RPC transaction methods in RFC 7047. Following RPC methods are supported in OVSDB protocol.:

- List databases
- Get schema
- Transact

- Cancel
- Monitor
- Update notification
- Monitor cancellation
- Lock operations
- Locked notification
- Stolen notification
- Echo

According to RFC 7047, an OVSDB server must implement all methods, and an OVSDB client is only required to implement the "Echo" method and otherwise free to implement whichever methods suit its needs. However, OVSDB library currently doesn't support all RPC methods. For "echo" method, the library can handle "echo" message from peer and send JSON response message back, but the library doesn't support actively sending "echo" JSON request to peer. Other unsupported RPC methods are listed below:

- Cancel
- Lock operations
- Locked notification
- Stolen notification

In OVSDB library the RPC methods are defined in Java interface OvsdbRPC. The library also provides a high-level interface OvsdbClient as main interface to interact with peer through OVSDB protocol. In passive connection scenario, each connection will have a corresponding OvsdbClient object, and the application can obtain the OvsdbClient object through connection listener callback methods. In other words, if the application implements OvsdbConnectionListener interface, it will get notification of connection status change with corresponding OvsdbClient object of that connection.

OVSDB database operations

RFC 7047 also defines database operations, such as insert, delete, and update, to be performed as part of a "transact" RPC request. OVSDB library defines the data operations in Operations.java and provides TransactionBuilder class to help build "transact" RPC request. To build a JSON-RPC transact request message, the application can obtain TransactionBuilder object through transactBuilder() method in OvsdbClient interface.

The TransactionBuilder class provides following methods to help build transaction:

- getOperations(): Get the list of operations in this transaction.
- add(): Add data operation to this transaction.
- build(): Return the list of operations in this transaction. Same as getOpeartions() method.
- execute(): Send the JSON RPC transaction to peer.
- getDatabaseSchema(): Get the database schema of this transaction.

If the application wants to build and send a "transact" RPC request to modify OVSDB tables on peer, it can do following steps: 1. Statically import parameter "op" in Operations.java

```
import static org.opendaylight.ovsdb.lib.operations.Operations.op;
```

2. Obtain transaction builder through transacBuilder() method in OvsdbClient:

```
TransactionBuilder transactionBuilder = ovsdbClient.transactionBuilder(dbSchema);
```

3. Add operations to transaction builder:

```
transactionBuilder.add(op.insert(schema, row));
```

4. Send transaction to peer and get JSON RPC response:

```
operationResults = transactionBuilder.execute().get();
```

Note: Although "select" operation is supported in OVSDB library, the library implementation is a little different from RFC 7047. In RFC 7047, section 5.2.2 describes "select" operation as follows:

"The "rows" member of the result is an array of objects. Each object corresponds to a matching row, with each column specified in "columns" as a member, the column's name as the member name, and its value as the member value. If "columns" is not specified, all the table's columns are included (including the internally generated "_uuid" and "_version" columns)."

The OVSDB library implementation always requires the column's name in "columns" field of JSON message. If the "columns" field is not specified, none of the table's columns are included. If the application wants to get the table entry with all columns, it needs to specify all columns' names to "columns" field.

Reference Documentation

RFC 7047 The Open vSwitch Databse Management Protocol https://tools.ietf.org/html/rfc7047

Contact

Hsin-Yi Shen syshen66@gmail.com