Table of Contents

Message Sink Simulator Implementation

1. Logging all the **incoming** and **outgoing** message
2. All the **outgoing** message will be **ByPass**
3. All the **incoming** message can be handled by **ByPass**, **Hold**, **Drop**, **Delay**, depending the rules specified.

The reason we **ByPass** all the outgoing message was due to the limitation of **Message** class, which doesn't include the message is targeting at.

How It Works?

You can assign **SinkControlState** to **MessageSink**, which comprises of a set of **Rules**, and a **DefaultRule**. Each **Rule** comes with

- **MessageMatcher**: specifies how to match the incoming message.
  1. **FromEndPoint**: Matching by the **From** endpoint.
  2. **Verb**: Matching by **Verb**
- **MessageHandler**: we provided four handlers
  1. **ByPassMessageHandler**: Just accept the message normally
  2. **DropMessageHandler**: Drop it, like it was lost
  3. **HoldMessageHandler**: Transfer the message into **SinkControlQueue**, which you can relase or drop it from queue later.
  4. **DelayMessageHandler**: It will hold the message, and release it later when specified milliseconds due.

As **DefaultRule**, it only contains **MessageHandler**, and no **MessageMatcher** part, which means it will handle all the message which not matched by the other rules.

⚠ Note: By default, **DefaultRule** is set to **ByPassMessageHandler**, unless you change it by calling **setDefaultRule()** to change it.

Let's say there is one **incoming** message, will be matched by the set of **Rules** sequentially. If it was matched one of **Rule**, then be handled by the associated **MessageHandler**. If none of **Rules** matched, then it will handled by the **DefaultRule**.

Sample code for assigning SinkControlState to node

To put every message received to Hold

```
SinkControlState holdAllState = new SinkControlState();
holdAllState.setDefaultRule(new HoldMessageHandler());
MessageSinkAPIs.setSinkControlState(targetNode, holdAllState);
```

To drop message which coming from node1 with verb XXX, and ByPass the other messages

```
SinkControlState dropSomeState = new SinkControlState();
dropSomeState.putControlRule(new MessageMatcher(node1, Verbs.XXX), new DropMessageHandler() );
MessageSinkAPIs.setSinkControlState(targetNode, dropSomeState);
```

A more complicated ruleset,

```
SinkControlState sct = new SinkControlState();
sct.putControlRule(new MessageMatcher(node2, Verbs.XXX), new HoldMessageHandler() );
sct.putControlRule(new MessageMatcher(node3), new DelayMessageHandler(20L) );
sct.setDefaultRule(new DropMessageHandler() );
MessageSinkAPIs.setSinkControlState(targetNode, sct);
```

To reset to default state, which is ByPass all the message

```
MessageSinkAPIs.resetSinkControlState(targetNode);
```

To reset all the nodes to default state, which is ByPass all the message

```
MessageSinkAPIs.resetSinkControlState(numberOfNodes);
```

Sample code for releasing or dropping messages which currently holding in SinkControlQueue

All the message which handled by **HoldMessageHandler**, will be stored in **SinkControlQueue**, which can be **release** or **drop** from queue later.

⚠ Note: By default, all messages stored in queue will be **release** when you assigning new **SinkControlState** to the **MessageSink**.

To release all the message from queue, which with verb XXX

```
MessageSinkAPIs.releaseFromQueue(tagetNode, new MessageMatcher(Verbs.XXX));
```

To release all the message from queue, which is from nodeX with verb XXX

```
MessageSinkAPIs.releaseFromQueue(tagetNode, new MessageMatcher(nodeX, Verbs.XXX));
```

To relase all the message from queue,

```
MessageSinkAPIs.releaseAllFromQueue(tagetNode));
```

To release all the message from queue, which with verb XXX

```
MessageSinkAPIs.dropFromQueue(tagetNode, new MessageMatcher(Verbs.XXX));
```

Log generated from MessageSink

We can follow the steps below to generate some sample logs genrated from **MessageSink**,

1. Remove all the running nodes. Just to start from scratch.

   ```
   ./anode.pl removeall
   ```
2. Create five new nodes.

   ```
   ./anode.pl -c 5 create run
   ```
3. Populates data by invoking cassandra-cli.

   ```
   cd /tmp/cassandra/node1/conf
        ./cassandra-cli
        > set Objects.facets['rowid2']['review']['revision'] = 'xxxxxx'
        > get Objects.facets['rowid2']['review']['revision']
   ```
4. To see all the logs generated from **MessageSink**.

   ```
   cd /tmp/cassandra
    grep -r SinkLogger node*/logs/*.log
   ```

You will the log message like below,

```
node1/logs/system.log: INFO [MESSAGE-SERIALIZER-POOL:1] 2009-08-17 18:28:45,844 SinkControlState.
   SinkLogger: STATE:[OUTGO], FROM:[192.168.10.1:7000], ID:[936], VERB:[ROW-MUTATION-VERB-HANDLER
node1/logs/system.log: INFO [MESSAGE-DESERIALIZER-POOL:1] 2009-08-17 18:28:45,921 SinkControlStat
   SinkLogger: STATE:[BYPASS], FROM:[192.168.10.4:7000], ID:[936], VERB:[RESPONSE]
node1/logs/system.log: INFO [MESSAGE-SERIALIZER-POOL:2] 2009-08-17 18:28:54,949 SinkControlState.
   SinkLogger: STATE:[OUTGO], FROM:[192.168.10.1:7000], ID:[961], VERB:[ROW-READ-VERB-HANDLER]
node1/logs/system.log: INFO [MESSAGE-DESERIALIZER-POOL:2] 2009-08-17 18:28:54,972 SinkControlStat
   SinkLogger: STATE:[BYPASS], FROM:[192.168.10.4:7000], ID:[961], VERB:[RESPONSE]
node4/logs/system.log: INFO [MESSAGE-DESERIALIZER-POOL:1] 2009-08-17 18:28:45,871 SinkControlStat
   SinkLogger: STATE:[BYPASS], FROM:[192.168.10.1:7000], ID:[936], VERB:[ROW-MUTATION-VERB-HANDLE
node4/logs/system.log: INFO [MESSAGE-SERIALIZER-POOL:1] 2009-08-17 18:28:45,911 SinkControlState.
   SinkLogger: STATE:[OUTGO], FROM:[192.168.10.4:7000], ID:[936], VERB:[RESPONSE]
node4/logs/system.log: INFO [MESSAGE-DESERIALIZER-POOL:2] 2009-08-17 18:28:54,951 SinkControlStat
   SinkLogger: STATE:[BYPASS], FROM:[192.168.10.1:7000], ID:[961], VERB:[ROW-READ-VERB-HANDLER]
node4/logs/system.log: INFO [MESSAGE-SERIALIZER-POOL:2] 2009-08-17 18:28:54,969 SinkControlState.
   SinkLogger: STATE:[OUTGO], FROM:[192.168.10.4:7000], ID:[961], VERB:[RESPONSE]
```

For the writing part, the message can be analyzed like,

```
# Message sedning from node1-->node4, with VERB:[ROW-MUTATION-VERB-HANDLER]
node1/logs/system.log: INFO [MESSAGE-SERIALIZER-POOL:1] 2009-08-17 18:28:45,844 SinkControlState.
   SinkLogger: STATE:[OUTGO], FROM:[192.168.10.1:7000], ID:[936], VERB:[ROW-MUTATION-VERB-HANDLER
node4/logs/system.log: INFO [MESSAGE-DESERIALIZER-POOL:1] 2009-08-17 18:28:45,871 SinkControlStat
   SinkLogger: STATE:[BYPASS], FROM:[192.168.10.1:7000], ID:[936], VERB:[ROW-MUTATION-VERB-HANDLE

# After node4 processed the request, it send back *RESPONSE* message from node4--> node1
node4/logs/system.log: INFO [MESSAGE-SERIALIZER-POOL:1] 2009-08-17 18:28:45,911 SinkControlState.
   SinkLogger: STATE:[OUTGO], FROM:[192.168.10.4:7000], ID:[936], VERB:[RESPONSE]
node1/logs/system.log: INFO [MESSAGE-DESERIALIZER-POOL:1] 2009-08-17 18:28:45,921 SinkControlStat
   SinkLogger: STATE:[BYPASS], FROM:[192.168.10.4:7000], ID:[936], VERB:[RESPONSE]
```

For the reading part,

```
# Message sedning from node1-->node4, with VERB:[ROW-READ-VERB-HANDLER]
node1/logs/system.log: INFO [MESSAGE-SERIALIZER-POOL:2] 2009-08-17 18:28:54,949 SinkControlState.
   SinkLogger: STATE:[OUTGO], FROM:[192.168.10.1:7000], ID:[961], VERB:[ROW-READ-VERB-HANDLER]
node4/logs/system.log: INFO [MESSAGE-DESERIALIZER-POOL:2] 2009-08-17 18:28:54,951 SinkControlStat
   SinkLogger: STATE:[BYPASS], FROM:[192.168.10.1:7000], ID:[961], VERB:[ROW-READ-VERB-HANDLER]
```

```
node4/logs/system.log: INFO [MESSAGE-SERIALIZER-POOL:2] 2009-08-17 18:28:54,969 SinkControlState.
   SinkLogger: STATE:[OUTGO], FROM:[192.168.10.4:7000], ID:[961], VERB:[RESPONSE]
node1/logs/system.log: INFO [MESSAGE-DESERIALIZER-POOL:2] 2009-08-17 18:28:54,972 SinkControlStat
   SinkLogger: STATE:[BYPASS], FROM:[192.168.10.4:7000], ID:[961], VERB:[RESPONSE]
```

Suggestion on how MessageSink can be used to replay the message

Let's say we want to replay the writing part in the previous scenario,

1. To remove all the existing nodes, and create 5 new virtual nodes,

   ```
   NodeOpAPIs.removeAll();
   NodeOpAPIs.creatNodes(5);
   ```
2. Put every node to **HoldMessageHandler**,

   ```
   SinkControlState holdAllState = new SinkControlState();
   holdAllState.setDefaultRule(new HoldMessageHandler());
   MessageSinkAPIs.setSinkControlState(5, holdAllState);
   ```
3. Query node4 whether **SinkControlQueue** contains a message from node1 with verb ROW-MUTATION-VERB-HANDLER

   ```
   MessageSinkAPIs.queryFromQuery(node4, new MessageMatcher(node1, Verbs.ROW-MUTATION-VERB
   ```
4. If it contains the message, release it

   ```
   MessageSinkAPIs.releaseFromQuery(node4, new MessageMatcher(node1, Verbs.ROW-MUTATION-VE
   ```
5. ...

-- ShangyYahoo -- 17 Aug 2009

Last revised by: shangy on 18 Aug 2009

ATTACHMENT LIST
%ATTACHMENTLIST{format="$fileName?"}%