How to use the Broker Java Admin API

Tools

Added by Robert Castaneda, last edited by louis@zezeran.com on Dec 03, 2009

The following example code connects to the broker and lists some statistics.

Note that a user should be part of the client group "admin" for this to work.

```
import COM.activesw.api.client.BrokerAdminClient;
import COM.activesw.api.client.BrokerClient;
import COM.activesw.api.client.BrokerException;
import COM.activesw.api.client.BrokerEvent;

public class BrokerAdminExample {

    /**
     * @param args
     * @throws BrokerException
     */
    public static void main(String[] args) throws BrokerException {
        BrokerAdminClient adminClient = new BrokerAdminClient("localhost:9949"
                , "RobsBroker"
                , null
                , "admin"
                ,"Broker Admin"
                , null );
            BrokerEvent stats = adminClient.getBrokerStats();
            System.out.println("Number of Events Published=" + stats.getIntegerField("numEventsPublished"));
            System.out.println("Documents Queued =" + stats.getIntegerField("traceNumEventsQueued"));





    }

}
```

The next example connects to the broker, locks a queue and then inspects all the items in that queue. Optional commented out code also deletes the item from the queue.

```
import COM.activesw.api.client.*;

public class BrokerAdmin
{
    public static void main(String[] args) throws BrokerException
    {
        String broker_host = "localhost:6849";
        String broker_name = null; // null means the default broker. One of only few uses for the setting
"Default Broker"
        String client_group = "admin";
        String admin_client_id = null;
        long queueLength = 0;
        /*
            This is the name of the client, which is basically your trigger.
            To find this out, make a trigger and then goto the MWS and Messaging
            Under Brokers -> Clients find the entry for your trigger (probably at the end of the list)
            Use this name below. You can tell you have the right one because it matches the format
            See how it has "Training" in it? Thats the IS's Client Prefix
        */
        String trigger_client_id ="Training_louis.friday_simptrig";

        BrokerLockedClientQueueBrowser browserQ = null;
        BrokerEvent[] browserEvents = null;
        BrokerEvent myStats = null;

        BrokerAdminClient c;
        BrokerConnectionDescriptor bcd;

        try
        {
            bcd = new BrokerConnectionDescriptor();
            c =  new BrokerAdminClient(broker_host, broker_name, admin_client_id,
                                 client_group, "AdminModifyClient", bcd);
        }
        catch (BrokerException ex)
        {
            System.out.println("Error on create admin client\n"+ex);
            return;
        }

        try
        {

            browserQ = c.createLockedClientQueueBrowser(trigger_client_id);
            myStats = c.getClientStatsById(trigger_client_id);
            queueLength = myStats.getLongField("queueLength");

            // Print out the Q length
            System.out.println("Length Of Queue Is: "+ queueLength);

            /*
                This time show the number of unacked events. If a document has been taken down to the client
                (Integration Server) but hasnt been successfully processed yet then it is unacked)
                You can test this by closing off the triggers Document Retrieval Tread (DRT) and
                Document Processing Thread (DPT) Set the local queue size to 1 and then open the DRT.
                1 document will be brought down from the Broker but not processed because the DPT is off.
            */

            System.out.println("Number of UnAcked Events: " + myStats.getIntegerField("numEventsUnacked"));
            browserEvents = browserQ.browseEvents((int) queueLength, 1000); // peek at events
            System.out.println(myStats.toString());

            for(int i=0;i<browserEvents.length;i++)
            {
                // My document had a field "name" so this works
                System.out.println(""+i+" "+browserEvents[i].getStringField("name"));
                System.out.println("Redelivery Count: " + browserEvents[i].getRedeliveryCount());
                System.out.println("Retrieved Status: " + browserEvents[i].getRetrievedStatus());
                // This line prints out the documents contents. Remember the broker stores documents as flat
strings
                System.out.println(browserEvents[i].toString());

                // Uncomment these lines to delete the items from the queue. More efficient would be to
define one
                // long array and then do one delete at the end.

                //long[] seqNums = new long[1];
                //seqNums[0] = browserEvents[i].getReceiptSequenceNumber();
                //browserQ.deleteEvents(seqNums);

            }
        }
        catch (Exception ex)
        {
            System.out.println("Error on changing queue\n"+ex);
            ex.printStackTrace();
        }
        finally
        {
            try
            {

                 browserQ.closeQueueBrowser();
            }
            catch (Exception ex)
            {
                System.out.println("Error on closing Q\n"+ex);
            }
        }
    }
}
```

See Also How to use the Broker Java API to Publish Documents

broker java admin api