A Short Tutorial on Java FTB Client Package v0.5

1. Introduction
   Java FTB Client Package provides API for Java programmers to write a FTB client code. A Java client, like any C-based FTB client, communicates with the FTB database server and a FTB agent to publish and subscribe to events through Fault Tolerant Backplane. Java FTB Client Package faithfully conforms to FTB API v0.5; it offers all equivalent API of C FTB 0.5, and thus allows an Java FTB client exchange events not only with other Java FTB clients but also with C FTB clients.

   Note: Java FTB Client Package is compatible with jdk 1.5 or higher.

2. Using Java FTB Client Package
   Java FTB Client Package comes in a single jar file, **jftb.jar.** To use the package, modify your shared library path to include **jftb.jar**. For example, for **bash** shell users, if **jftb.jar** is stored under /home/hoony/jftb, your shared library path should be changed like,

   % export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/hoony/jftb/jftb.jar

   Alternatively, users may inlcude the file path to ftb.jar during compilation or run time, i.e.

   % javac –cp /home/hoony/jftb/jftb.jar myJavaClient.java
   % java –cp /home/hoony/jftb/jftb.jar myJavaClient

   Java FTB Client Package assumes an enviornment variable **FTB_BSTRAP_SERVER** is set to include the IP address of the FTB database server. Otherwise, it assumes the FTB database server is running locally (i.e. **localhost**).

   To use Java FTB Client APIs, simply import ***ftb.client*** package*.*

   ```
   import ftb.client.*;

   public class myJavaClient
   {
       ...
   }
   ```

3. FTB_World: A Portal to Fault Tolerant Backplane.
   Prior to publishing or subscribing to events, a Java client first needs to create an instance of FTB_World. FTB_World is a portal to Fault Tolerant Backplane; eight of nine FTB APIs are available through FTB_World. If an instance of FTB_World already exists, it can be shared between clients. An instance of FTB_World is simply created by,

   ```
   FTB_World ftb = new FTB_World();
   ```

   The eight methods that FTB_World provides are:

- ***connect();***
- ***disconnect();***
- ***subscribe();***
- ***unsubscribe();***
- ***publish();***
- ***compare_handles();***
- ***declare_publishable_event();***
- ***get_event_handle();***

Only method that is not provided by FTB_World is ***poll()***. It is available through FTB_Subscription (See Subscription section for details).

4. Client Connection to FTB_World.

4.1 Client Information

A FTB client needs to connect to FTB_World before publishing or subscribing to events. A connection is made through the ***connect()*** method of ***FTB_World***. For a connection, a client should provide client information that includes,

- client event space
- client name
- client job ID
- client subscription style

Among these, only the "client event space" is mandatory, and the rest may be omitted. The class ***FTB_Client_Info*** serves as the placeholder for client information. Each attribute of the client information is loaded into FTB_Client_Info by its member method, ***set_attribute()***.

```
FTB_Client_Info client_info = new FTB_Client_Info();
client_info.set_attribute(FTB_Client_Info.FTB_CLIENT_EVENTSPACE,
"FTB.FTB_EXAMPLE.SIMPLE");
```

Optional information attribute can also be set using the same ***set_attribute()***, i.e.

```
set_attribute(FTB_Client_Info.FTB_CLIENT_NAME, "name");
set_atrribute(FTB_Client_Info.FTB_CLIENT_JOBID, "jobid");
set_attribute(FTB_Client_Info.FTB_CLIENT_SUBSCRIPTION_STYLE, "style");
```

4.2 Connection to FTB_World and Client Handle

A FTB client is identified by a client handle; a client uses its client handle to publish or subscribe to events. A client handle is assigned by FTB_World when the client registers with it (i.e. via ***connect()***). The client handle is also used to disconnect from FTB_World.

```
FTB_Client_Handle client_handle;
try {
    client_handle = ftb.connect(client_info);
}
catch (FTB_Exception fe) {
    ...
}
```

...
ftb.disconnect(client_handle);

5. Event Publishing

5.1 Event Information

To publish an event (or events), the type and definition of the event should be registered with the FTB in advance. This is accomplished in two steps. First, a client composes information about the event it wishes to publish. The information includes two attributes, event name and its level of severity. Second, the information is passed to the FTB for registration. Java Client Package provides the class ***FTB_Event_Info*** to construct event information. ***FTB_Event_Info*** is instantiated with event name and its severity as arguments. Registration of an event is done via ***decalare_publishable_event()*** of FTB_World.

```
FTB_Event_Info event_info = new FTB_Event_Info("SIMPLE_EVENT", "INFO");
ftb.decalare_publishable_event(client_handle, event_info);
```

If a client wishes to publish more than one event type, it needs to register each event type separately.

5.2 Event Publishing

A client publishes an event using ***publish()*** of ***FTB_World.*** To send out "SIMPLE_EVENT" event,

```
try {
    FTB_Event_Handle event_handle =
            ftb.publish(client_handle, "SIMPLE_EVENT", null);
}
```

5.3 Sending Payload Data

If a client wants to send out payload data with an event, ***FTB_Event_Property*** that contains payload data should be passed to ***publish()*** as an argument. By definition, payload data is a byte array of a fixed size (See FTB_DEF for details). Thus, to send out an array byte as the payload content,

```
FTB_Event_Property event_property = new FTB_Event_Property();
data[] = new byte[FTB_DEF.FTB_MAX_PAYLOAD_DATA];
  ...
event_property.set_paylod(data);

try {
    FTB_Event_Handle event_handle =
            ftb.publish(client_handle, "SIMPLE_EVENT", event_property);
}
```

Often, FTB clients use payload to exchange messages of strings. Since C-based clients expect 8 bit character array for a string, ***FTB_Event_Property*** converts Java String into

ASCII string and put it in the payload field. The same ***set_payload()*** is used for this purpose.

```
String message = "Hello World";
event_property.set_payload(message);
    ...
```

6. Event Subscription
   A client subscribes to an event with either **polling** or **callback** option. For both options, an event is subscribed through ***subscribe()*** method of FTB_World. ***subscribe()*** takes in four arguments: client handle, event subscription string, callback interface, and callback argument. If callback interface is set to **null**, the event is subscribed with polling option for default.

   6.1 Event Subscription with Callback
   The C counterpart API of ***connect()*** takes in the pointer to a callback function as an argument. Since Java does not directly support callback (notion of pointer is not supported), Java FTB Package utilizes ***Interface.*** That is, any client that wishes a callback to be invoked on receiving events, should implements ***FTB_Callable*** interface.

```
public interface FTB_Callable
{
    public abstract void message_received(FTB_Receive_Event event, Object obj);
}
```

More specifically, a client that wishes to associate a callback with an event needs to implement ***message_received(FTB_Receive_Event event, Object obj)***, where *event* is the event received (equivalent to ftb_receive_event_t in C APIs) and *obj* is a user-specified argument to the callback.

```
public class SimpleCallback implements FTB_Callable
{
    public void message_received(FTB_Receive_Event event, Object obj)
    {
            ...
    }

    public static void main(String args[])
    {
            SimpleCallback simple_subscriber =new SimpleCallback();
            FTB_World ftb = new FTB_World();
            ...
            FTB_Subscription subscription = null;
            try {
                    subscription = ftb.subscribe(client_handle,
                                                 "event_name=SIMPLE_EVENT",
                                                  simple_subscriber, null);
            }
            ...
    }
```

}

Note that ***client_handle*** is a ***FTB_Client_Handle***.

6.2 Event Subscription with Polling
For polling, we need to replace ***ftb.subscribe(…)*** with
ftb.subscribe(client_handle, "event_name=SIMPLE_EVENT", null, null);

An event subscription is denoted by ***FTB_Subscription,*** from which events are polled (this is one different aspect from C APIs).

FTB_Receive_Event event = subscription.poll();

6.3 Event Unsubscription
A subscription to an event is cancelled via ***unsubscribe()*** of ***FTB_World.***

ftb.unsubscribe(subscription);

7. Comparing Event Handles.
An event handle is associated with an outgoing or incoming event. Event handles are used to compare whether two events are identical. For every outgoing event, an event handle is returned by ***publish()*** (See Section 4.2). The event handle of an incoming event (i.e. FTB_Receive_Event) is obtained via ***get_event_handle().***

FTB_Receive_Event event = subscription.poll();
FTB_Event_Handle event_handle = ftb.get_event_handle(event);

compare_event_handles() compares two event handles and returns **true** if they denote the same event.

```
if ( ftb.compare_event_handles(evt_hdl1, evt_hdl2) ) {
    System.out.println("They are identical event handles");
}
```

8. Exception
Java FTB Client Package defines an ***FTB_Exception*** to denote the error code from three methods.
   - ***connect()***
   - ***publish()***
   - ***subscribe()***

These methods that are expected to return some handle objects upon success throw ***FTB_Exception*** when their execution fail. The thrown FTB_Exception will include error code of the failed operation. The details of error codes are listed in FTB_DEF.