```
/**
* The <code>subscribeFloat</code> method is used to subscribe to a status variable of type float. The leaf QoS broker will set up an path
* so that we will receive the latest value of the variable.
* <BR>
* @param publisherName The name of the publisher.
* @param variableName The name of the variable to be subscribed to.
* @param modes The modes that this subscription will be valid in.
* @param interval The interval that we wish to receive the events.
* @param priority The priority of the subscription.
* @param latency The latency that we would like for this subscription.
* @param redundancy The redundancy that we would like for this subscription.
* @param holderObject Where the subscribed to value will be placed.
* @return Returns true if message is subscribed to, false otherwise.
*/
public short subscribeFloat(String publisherName, String variableName, int modes, int[] interval, short[] priority,
                            short[] latency, short[] redundancy, HolderFloatInterface holderObject);

/**
* The <code>subscribeBoolean</code> method is used to subscribe to a status variable of type boolean. The leaf QoS broker will set up
* an path so that we will receive the latest value of the variable.
* <BR>
* @param publisherName The name of the publisher.
* @param variableName The name of the variable to be subscribed to.
* @param modes The modes that this subscription will be valid in.
* @param interval The interval that we wish to receive the events.
* @param priority The priority of the subscription.
* @param latency The latency that we would like for this subscription.
* @param redundancy The redundancy that we would like for this subscription.
* @param holderObject Where the subscribed to value will be placed.
* @return Returns true if message is subscribed to, false otherwise.
*/
public short subscribeBoolean(String publisherName, String variableName, int modes, int[] interval, short[] priority,
                              short[] latency, short[] redundancy, HolderBooleanInterface holderObject);

/**
* The <code>subscribeUserDefined</code> method is used to subscribe to a status variable of type boolean. The leaf QoS broker will set
* up an path so that we will receive the latest value of the variable.
* <BR>
* @param publisherName The name of the publisher.
* @param variableName The name of the variable to be subscribed to.
* @param modes The modes that this subscription will be valid in.
* @param interval The interval that we wish to receive the events.
* @param priority The priority of the subscription.
* @param latency The latency that we would like for this subscription.
* @param redundancy The redundancy that we would like for this subscription.
* @param holderObject Where the subscribed to value will be placed.
* @param userType The type of the user registered type.
* @return Returns true if message is subscribed to, false otherwise.
*/
public short subscribeUserDefined(String publisherName, String variableName, int modes, int[] interval, short[] priority,
                                  short[] latency, short[] redundancy, HolderUserDefinedInterface holderObject, int userType);

/**
* The <code>unSubscribe</code> method is used to unsubscribe from one of the variables that we have subscribed to.
* <BR>
* @param publisherName The name of the publisher.
* @param variableName The name of the variable subscribed to.
* @return Returns <code>0</code> if message is unSubscribed to, error code
* otherwise.
*/
public short unSubscribe(String publisherName, String variableName);

/**
* The <code>unSubscribeAll</code> method is used to remove all the subscriptions.
* <BR>
* @return returns <code>0</code> of all the subscriptions is removed,
* error code otherwise.
*/
public short unSubscribeAll();

/**
* The <code>subscribeAll</code> method is used to subscribe to all flooding alerts that we will receive.
* <BR>
* @param floodedVariables This is where we will store all the flooded values that we will receive
* @param floodedReceived Uses this one to signal the user that a event has been recveived.
*     NOTE 1: if you don't want to be signaled then give <code>null</code> for this argument.
*     NOTE 2: The type that is signaled to you is <code>HolderUnknown</code>.
*/
public void subscribeAllFlooding(IntHashMap floodedVariables, ArrayBlockingQueue floodedReceived);

/**
* The <code>unSubscribeAllFlooding</code> method is used to unsubscribe from
* receiving all the flooded events.
*/
public void unSubscribeAllFlooding();
```