```
  /**
  * The <code>createGroupSubscription</code> method is used to subscribe to a status variable of type int that will be added to a group.
  * The leaf QoS broker will set up an path so that we will receive the latest value of the variable.
  * <BR>
  * @param groupName The name of the group to be created.
  * @param history How many values should be held for each variable in the history.
  * @param modes The modes that this subscription will be valid in.
  * @param subInterval The interval that we wish to receive the events.
  * @param priority The priority of the subscription.
  * @return Returns the created group, null if it already existed.
  */
  public HolderBaseGroupInterface createGroupSubscription(String groupName, int history, int modes, int[] subInterval, short[] priority);

  /**
  * The <code>removeGroupSubscription</code> method is used to remove one of the group subscriptions that we have.
  * <BR>
  * @param groupName The name of the group to be removed.
  * @return returns <code>true</code> if the subscription is removed, <code>false</code> otherwise.
  */
  public boolean removeGroupSubscription(String groupName);

  /**
  * The <code>subscribeIntForGroup</code> method is used to subscribe to a status variable of type int that will be added to a group.
  * The leaf QoS broker will set up an path so that we will receive the latest value of the variable.
  * <BR>
  * @param groupName The name of the group where the subscription will belong.
  * @param publisherName The name of the publisher.
  * @param variableName The name of the variable to be subscribed to.
  * @param latency The latency that we would like for this subscription.
  * @param redundancy The redundancy that we would like for this subscription.
  * @return Returns true if message is subscribed to, false otherwise.
  */
  public short subscribeIntForGroup(String groupName, String publisherName, String variableName, short[] latency, short[] redundancy);

  /**
  * The <code>subscribeFloatForGroup</code> method is used to subscribe to a status variable of type float that will be added to a group.
  * The leaf QoS broker will set up an path so that we will receive the latest value of the variable.
  * <BR>
  * @param groupName The name of the group where the subscription will belong.
  * @param publisherName The name of the publisher.
  * @param variableName The name of the variable to be subscribed to.
  * @param latency The latency that we would like for this subscription.
  * @param redundancy The redundancy that we would like for this subscription.
  * @return Returns true if message is subscribed to, false otherwise.
  */
  public short subscribeFloatForGroup(String groupName, String publisherName, String variableName, short[] latency, short[] redundancy);

  /**
  * The <code>subscribeBooleanForGroup</code> method is used to subscribe to a status variable of type boolean that will be added to a
  * group. The leaf QoS broker will set up an path so that we will receive the latest value of the variable.
  * <BR>
  * @param groupName The name of the group where the subscription will belong.
  * @param publisherName The name of the publisher.
  * @param variableName The name of the variable to be subscribed to.
  * @param latency The latency that we would like for this subscription.
  * @param redundancy The redundancy that we would like for this subscription.
  * @return Returns true if message is subscribed to, false otherwise.
  */
  public short subscribeBooleanForGroup(String groupName, String publisherName, String variableName, short[] latency, short[] redundancy);

  /**
  * The <code>subscribeUserDefinedForGroup</code> method is used to subscribe to a status variable of type UserDefined that will be added
  * to a group. The leaf QoS broker will set up an path so that we will receive the latest value of the variable.
  * <BR>
  * @param groupName The name of the group where the subscription will belong.
  * @param publisherName The name of the publisher.
  * @param variableName The name of the variable to be subscribed to.
  * @param latency The latency that we would like for this subscription.
  * @param redundancy The redundancy that we would like for this subscription.
  * @param holderObject The object of user defined type where the values will be stored.
  * @param userType The type of the object that the user have defined.
  * @return Returns true if message is subscribed to, false otherwise.
  */
  public short subscribeUserDefinedForGroup(String groupName, String publisherName, String variableName, short[] latency,
                                            short[] redundancy, HolderUserDefinedInterface holderObject, int userType);

  /**
  * The <code>unSubscribeForGroup</code> method is used to unsubscribe from one of the variables that we been subscribed to for this group.
  * <BR>
  * @param groupName The name of the group where the subscription belong.
  * @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 unSubscribeForGroup(String groupName, String publisherName, String variableName);
}
```