```
/**
* The <code>publish</code> method is used to publish a float event.
* <BR>
* @param variableId The unique Id of this status variable
* @param value The value to be published.
* @param timeStamp The timestamp that this event will get.
* @return Returns <code>true</code> if message is published, <code>false</code> otherwise.
*/
public boolean publish(int variableId, float value, long timeStamp);

/**
* The <code>publish</code> method is used to publish a boolean event.
* <BR>
* @param variableId The unique Id of this status variable
* @param value The value to be published.
* @return Returns <code>true</code> if message is published, <code>false</code> otherwise.
*/
public boolean publish(int variableId, boolean value);

/**
* The <code>publish</code> method is used to publish a boolean event.
* <BR>
* @param variableId The unique Id of this status variable
* @param value The value to be published.
* @param timeStamp The timestamp that this event will get.
* @return Returns <code>true</code> if message is published, <code>false</code> otherwise.
*/
public boolean publish(int variableId, boolean value, long timeStamp);

/**
* The <code>registerPublishInt</code> method is to register a publish with the edgeStatusRouter.
* <BR>
* @param variableName The name of the variable which evente is to be published.
* @param priority The priority of this status variable.
* @param intervalLow The lowest interval that the message will be published.
* @param intervalHigh The highest interval that the message will be published.
* @return Returns the variableId of the registered publication.
* @throws PublisherException If something went wrong during the registration.
*/
public int registerPublishInt(String variableName, short priority, int intervalLow, int intervalHigh) throws PublisherException;

/**
* The <code>registerPublishFloat</code> method is to register a publish with the edgeStatusRouter.
* <BR>
* @param variableName The name of the variable which evente is to be published.
* @param priority The priority of this status variable.
* @param intervalLow The lowest interval that the message will be published.
* @param intervalHigh The highest interval that the message will be published.
* @return Returns the variableId of the registered publication.
* @throws PublisherException If something went wrong during the registration.
*/
public int registerPublishFloat(String variableName, short priority, int intervalLow, int intervalHigh) throws PublisherException;

/**
* The <code>registerPublishBoolean</code> method is to register a publish with the edgeStatusRouter.
* <BR>
* @param variableName The name of the variable which evente is to be published.
* @param priority The priority of this status variable.
* @param intervalLow The lowest interval that the message will be published.
* @param intervalHigh The highest interval that the message will be published.
* @return Returns the variableId of the registered publication.
* @throws PublisherException If something went wrong during the registration.
*/
public int registerPublishBoolean(String variableName, short priority, int intervalLow, int intervalHigh) throws PublisherException;

/**
* The <code>registerPublishUserDefined</code> method is to register a publish with the edgeStatusRouter.
* <BR>
* @param variableName The name of the variable which evente is to be published.
* @param priority The priority of this status variable.
* @param intervalLow The lowest interval that the message will be published.
* @param intervalHigh The highest interval that the message will be published.
* @param userType The type of the user registered type.
* @param length The length iin bytes of the user defined type.
* @return Returns the variableId of the registered publication.
* @throws PublisherException If something went wrong during the registration.
*/
public int registerPublishUserDefined(String variableName, short priority, int intervalLow,
                                      int intervalHigh, int userType, int length) throws PublisherException;

/**
* The <code>registerPublishIntAlert</code> method is to register a publish with the edgeStatusRouter.
* <BR>
* @param variableName The name of the variable which evente is to be published.
* @param minInterval The minimum interval between two alerts being published. This will
* prevent the publisher from flooding the network.
* @return Returns the variableId of the registered publication.
* @throws PublisherException If something went wrong during the registration.
*/
public int registerPublishIntAlert(String variableName, int minInterval) throws PublisherException;
```