# Association

Association is a protocol relationship between endpoints. Its wrapper over actual socket exposing the same API's irrespective if its client side socket initiating connection or server side socket accepting connection. Also the underlying scoket can be of type TCP or SCTP.

The Application using Restcomm SCTP Library calls management interface to create new instance of association and keeps reference to this instance for lifetime of association for seding the PayloadData.

The `Association.java` API looks like

```
package org.mobicents.protocols.api;

import io.netty.buffer.ByteBufAllocator;



/**
 * <p>
 * A protocol relationship between endpoints
 * </p>
 * <p>
 * The implementation of this interface is actual wrapper over Socket that
 * know's how to communicate with peer. The user of Association shouldn't care
 * if the underlying Socket is client or server side
 * </p>
 * <p>
 *
 * </p>
 *
 * @author amit bhayani
 *
 */
public interface Association {

    /**
     * Return the Association channel type TCP or SCTP
     *
     * @return
     */
    public IpChannelType getIpChannelType();

    /**
     * Return the type of Association CLIENT or SERVER
     *
     * @return
     */
    public AssociationType getAssociationType();

    /**
     * Each association has unique name
```

```
     *
     * @return name of association
     */
    public String getName();

    /**
     * If this association is started by management
     *
     * @return
     */
    public boolean isStarted();

    /**
     * If this association up (connection is started and established)
     *
     * @return
     */
    public boolean isConnected();

    /**
     * If this association up (connection is established)
     *
     * @return
     */
    public boolean isUp();

    /**
     * The AssociationListener set for this Association
     *
     * @return
     */
    public AssociationListener getAssociationListener();

    /**
     * The {@link AssociationListener} to be registered for this Association
     *
     * @param associationListener
     */
    public void setAssociationListener(AssociationListener associationListener);

    /**
     * The host address that underlying socket is bound to
     *
     * @return
     */
    public String getHostAddress();

    /**
     * The host port that underlying socket is bound to
     *
     * @return
```

```
     */
    public int getHostPort();

    /**
     * The peer address that the underlying socket connects to
     *
     * @return
     */
    public String getPeerAddress();

    /**
     * The peer port that the underlying socket is connected to
     *
     * @return
     */
    public int getPeerPort();

    /**
     * Server name if the association is for {@link Server}
     *
     * @return
     */
    public String getServerName();

    /**
     * When SCTP multi-homing configuration extra IP addresses are here
     *
     * @return
     */
    public String[] getExtraHostAddresses();

    /**
     * Send the {@link PayloadData} to the peer
     *
     * @param payloadData
     * @throws Exception
     */
    public void send(PayloadData payloadData) throws Exception;

    /**
     * Return ByteBufAllocator if the underlying Channel is netty or null if not
     *
     * @return
     */
    public ByteBufAllocator getByteBufAllocator() throws Exception;

    /**
     * Return the last measured Congestion Level at the sending direction
     *
     * @return
     */
```

```
    public int getCongestionLevel();

    /**
     * Use this method only for accepting anonymous connections
     * from the ServerListener.onNewRemoteConnection() invoking
     *
     * @param associationListener
     * @throws Exception
     */
    public void acceptAnonymousAssociation(AssociationListener associationListener)
throws Exception;

    /**
     * Use this method only for rejecting anonymous connections
     * from the ServerListener.onNewRemoteConnection() invoking
     */
    public void rejectAnonymousAssociation();

    /**
     * Stop the anonymous association. The connection will be closed and we will not
reuse this association
     * This can be applied only for anonymous association, other associations must be
stopped by
     * Management.stopAssociation(String assocName)
     *
     * @throws Exception
     */
    public void stopAnonymousAssociation() throws Exception;

}
```

Application interested in receiving payload from underlying socket registers the instance of class implementing AssociationListener with this Association.

The `AssociationListener.java` API looks like

```
package org.mobicents.protocols.api;

/**
 * <p>
 * The listener interface for receiving the underlying socket status and
 * received payload from peer. The class that is interested in receiving data
 * must implement this interface, and the object created with that class is
 * registered with {@link Association}
 * </p>
 *
 * @author amit bhayani
 *
 */
public interface AssociationListener {
```

```
    /**
     * Invoked when underlying socket is open and connection is established with
     * peer. This is expected behavior when management start's the
     * {@link Association}
     *
     * @param association
     * @param maxInboundStreams
     *            Returns the maximum number of inbound streams that this
     *            association supports. Data received on this association will
     *            be on stream number s, where 0 <= s < maxInboundStreams(). For
     *            TCP socket this value is always 1
     * @param maxOutboundStreams
     *            Returns the maximum number of outbound streams that this
     *            association supports. Data sent on this association must be on
     *            stream number s, where 0 <= s < maxOutboundStreams(). For TCP
     *            socket this value is always 1
     */
    public void onCommunicationUp(Association association, int maxInboundStreams, int
maxOutboundStreams);

    /**
     * Invoked when underlying socket is shutdown and connection is ended with
     * peer. This is expected behavior when management stop's the
     * {@link Association}
     *
     * @param association
     */
    public void onCommunicationShutdown(Association association);

    /**
     * Invoked when underlying socket lost the connection with peer due to any
     * reason like network between peer's died etc. This is unexpected behavior
     * and the underlying {@link Association} should try to re-establish the
     * connection
     *
     * @param association
     */
    public void onCommunicationLost(Association association);

    /**
     * Invoked when the connection with the peer re-started. This is specific to
     * SCTP protocol
     *
     * @param association
     */
    public void onCommunicationRestart(Association association);

    /**
     * Invoked when the {@link PayloadData} is received from peer
     *
```

```
     * @param association
     * @param payloadData
     */
    public void onPayload(Association association, PayloadData payloadData);

    /**
     * <p>
     * The stream id set in outgoing {@link PayloadData} is invalid. This packe
     * will be dropped after calling the listener.
     * </p>
     * <p>
     * This callback is on same Thread as {@link SelectorThread}. Do not delay
     * the process here as it will hold all other IO.
     * </p>
     *
     * @param payloadData
     */
    public void inValidStreamId(PayloadData payloadData);

}
```