Mobicents SCTP Library User Guide

by Amit Bhayani

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts* [https://fedorahosted.org/liberation-fonts/] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight key caps and key-combinations. For example:

> To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the `cat my_next_bestselling_novel` command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key cap, all presented in Mono-spaced Bold and all distinguishable thanks to context.

Key-combinations can be distinguished from key caps by the hyphen connecting each part of a key-combination. For example:

> Press **Enter** to execute the command.
>
> Press **Ctrl**+**Alt**+**F1** to switch to the first virtual terminal. Press **Ctrl**+**Alt**+**F7** to return to your X-Windows session.

The first sentence highlights the particular key cap to press. The second highlights two sets of three key caps, each set pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in `Mono-spaced Bold`. For example:

> File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialogue box text; labelled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System > Preferences > Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications > Accessories > Character Map** from the main menu bar. Next, choose **Search > Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit > Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in Proportional Bold and all distinguishable by context.

Note the **>** shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select **Mouse** from the **Preferences** sub-menu in the **System** menu of the main menu bar' approach.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether Mono-spaced Bold or Proportional Bold, the addition of Italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type `ssh` *`username@domain.name`* at a shell prompt. If the remote machine is `example.com` and your username on that machine is john, type `ssh john@example.com`.
>
> The `mount -o remount` *`file-system`* command remounts the named file system. For example, to remount the `/home` file system, the command is `mount -o remount /home`.
>
> To see the version of a currently installed package, use the `rpm -q` *`package`* command. It will return a result as follows: *`package-version-release`*.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as

a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules* (*MPMs*). Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

1.2. Pull-quote Conventions

Two, commonly multi-line, data types are set off visually from the surrounding text.

Output sent to a terminal is set in `Mono-spaced Roman` and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `Mono-spaced Roman` but are presented and highlighted as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }

}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

A note is a tip or shortcut or alternative approach to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring Important boxes won't cause data loss but may cause irritation and frustration.

Warning

A Warning should not be ignored. Ignoring warnings will most likely cause data loss.

2. Provide feedback to the authors!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in the the *Issue Tracker* [http://code.google.com/p/mobicents/issues/list], against the product **Mobicents SCTP Library** , or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: SCTPLibrary_User_Guide

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Introduction to Mobicents SCTP Library

In computer networking, the Stream Control Transmission Protocol *(SCTP)* [http://en.wikipedia.org/wiki/SCTP] is a Transport Layer protocol, serving in a similar role to the popular protocols Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). It provides some of the same service features of both: it is message-oriented like UDP and ensures reliable, in-sequence transport of messages with congestion control like TCP.

The protocol was defined by the IETF Signaling Transport (SIGTRAN) working group in 2000 and is maintained by the IETF Transport Area (TSVWG) working group. *RFC 4960* [http://tools.ietf.org/html/rfc4960] defines the protocol. *RFC 3286* [http://tools.ietf.org/html/rfc3286] provides an introduction.

Mobicents SCTP Library is providing the convenient API's over Java SCTP, hence can be used only with version JDK 1.7 or above.

In addition to exposing the SCTP protocol, Mobicents SCTP Library contains number of features which other wise the application depending on SCTP will have to take care of. For example Mobicents SCTP Library provides

- Management interface to manage the underlying SCTP Associations
- Persistence mechanism capable of initiating the SCTP Association if the system is restarted after crash or gracefull shutdown
- Tries re-initiate the connection if for some reason the connection is lost between the peers
- Configuration to make the module behave as single thread or multi-threaded depending on requirement's of application

Below diagram shows various layers involved

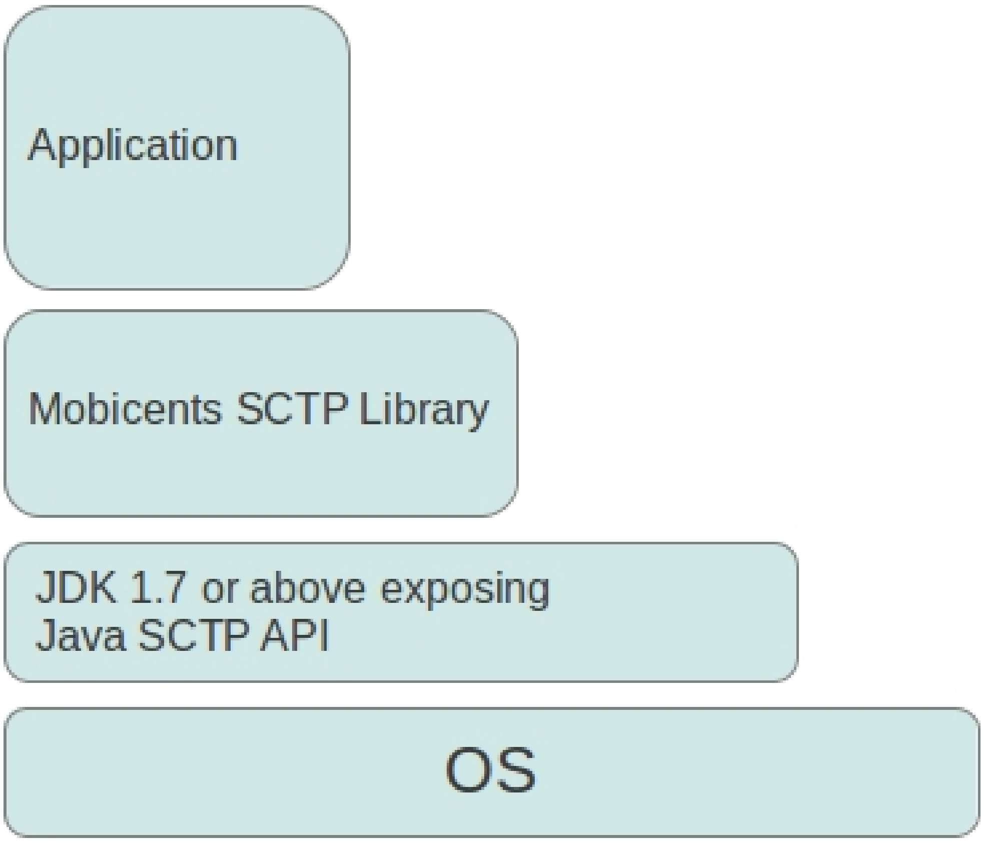

Figure 1.1. Layers involved

Setup

2.1. Mobicents SCTP Library Source Code

2.1.1. Release Source Code Building

1. **Downloading the source code**

   Important

   Subversion is used to manage its source code. Instructions for using Subversion, including install, can be found at *http://svnbook.red-bean.com*

   Use SVN to checkout a specific release source, the base URL is http://mobicents.googlecode.com/svn/tags/protocols/sctp, then add the specific release version, lets consider 1.0.0.CR1.

```
[usr]$ svn co http://mobicents.googlecode.com/svn/tags/protocols/sctp/1.0.0.CR1 sctp-1.0.0.CR1
```

2. **Building the source code**

   Important

   Maven 2.0.9 (or higher) is used to build the release. Instructions for using Maven2, including install, can be found at *http://maven.apache.org*

   Use Maven to build the binaries.

```
        [usr]$ cd sctp-1.0.0.CR1
        [usr]$ mvn install
```

   Once the process finishes you should have the `binary` jar files in the `target` directory of `module`.

2.1.2. Development Trunk Source Building

Similar process as for *Section 2.1.1, “Release Source Code Building”*, the only change is the SVN source code URL, which is http://mobicents.googlecode.com/svn/trunk/protocols/sctp.

Design Overview

The internal structure of Mobicents SCTP Library looks like

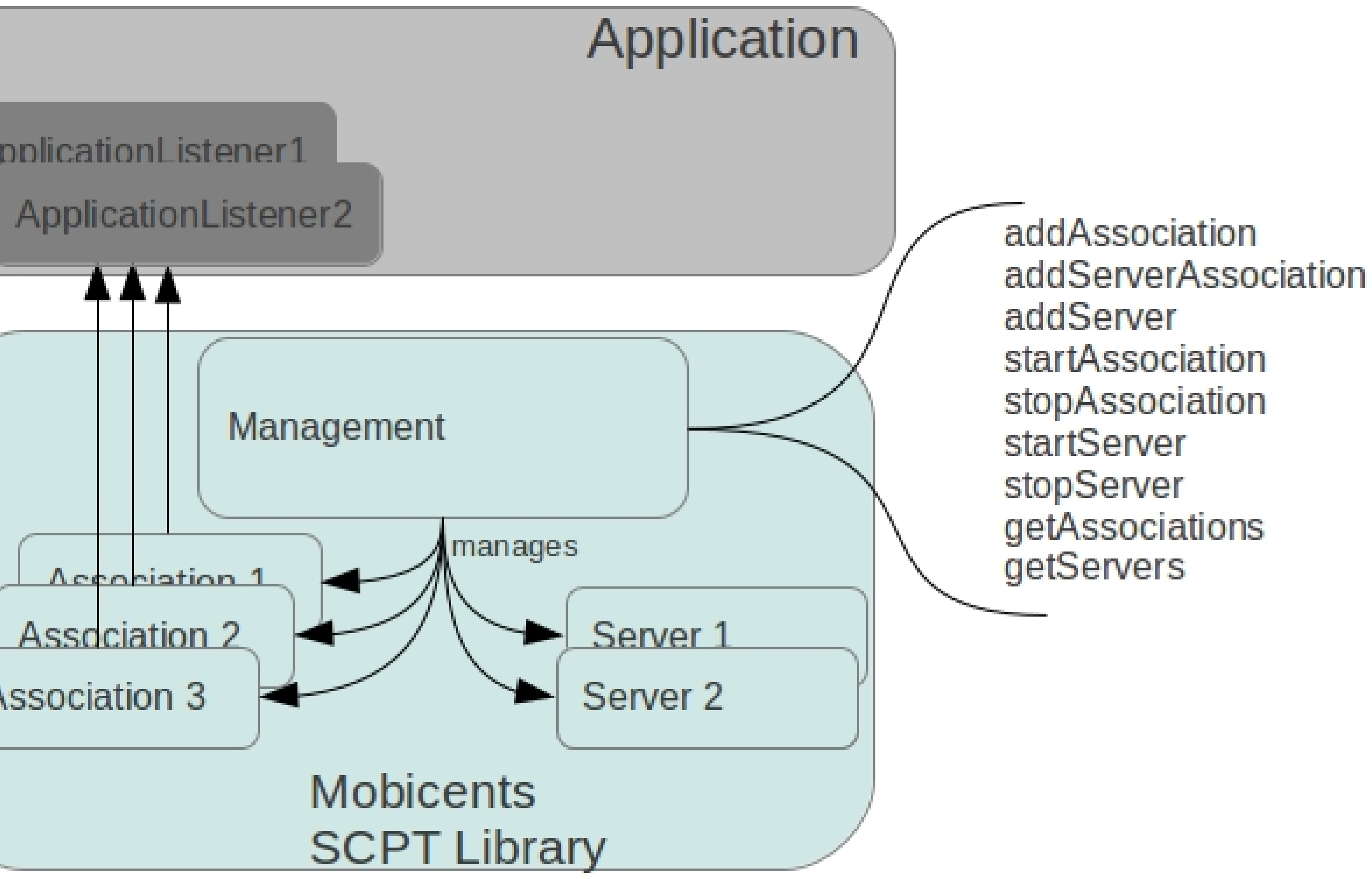

Figure 3.1. Architecture

The prime responsibility of Mobicents SCTP Library is abstract Application from underlying SCTP sockets and expose same API (`Association.java`) irrespective if the underlying SCTP is acting as server side waiting for client to connect or client side initiating connection.

The management (`Management.java`) controls the associations and servers. The Application can execute commands to create/delete associations/servers.

Management

In addition to managing the associations and servers, management also persists the state of each in XXX_sctp.xml file, where XXX is unique name given to management instance.

If there is system crash, management is responsible to bring the associations and servers back to same state it was before the crash. For example if client side association was connected to peer server before crash, management will try to connect back to peer server after restoration

4.1. API

The `Management.java` API looks like

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

import java.util.List;
import java.util.Map;

/**
 * <p>
 * {@link Management} class manages the underlying {@link Association} and
 * {@link Server}.
 * </p>
 * <p>
 * Management should persist the state of {@link Server} and {@link Association}
 * </p>
 * <p>
 * Management when {@link #start() started} looks for file <tt>XXX_sctp.xml</
tt> containing serialized state of underlying
 * {@link Association} and {@link Server}. Set the directory path by calling {@link #setPersistDir(String)} to direct M
 * directory for underlying serialized file.
 * </p>
 * <p>
 * If directory path is not set, Management searches for system property
 * <tt>sctp.persist.dir</tt> to get the path for directory
 * </p>
 * <p>
 * Even if <tt>sctp.persist.dir</tt> system property is not set,
 * Management will look at System set property <tt>user.dir</tt>
 * </p>
 *
 * @author amit bhayani
 *
 */
public interface Management {

    /**
     * Returns the name of this {@link Management} instance
     *
     * @return
     */
    public String getName();

    /**
     * Directory where the XXX.xml will be searched
     * @param persistDir
     */
    public void setPersistDir(String persistDir);

    /**
     * Start the management. No management operation can be executed unless
     * {@link Management} is started. If {@link Server} and {@link Association}
     * were defined previously, Management should recreate those {@link Server}
     * and {@link Association}.
     *
     * @throws Exception
     */
    public void start() throws Exception;

    /**
     * Stop the management. It should persist the state of {@link Server} and
     * {@link Associtaion}.
     *
     * @throws Exception
     */
    public void stop() throws Exception;

    /**
     * Add new {@link Server}.
     *
     * @param serverName
     *            name of the Server. Should be unique name
     * @param hostAddress
     *            IP address that this server will bind to
     * @param port
     *            port that this server will bind to
     * @return new Server instance
     * @throws Exception
     *             Exception if management not started or server name already
     *             taken or some other server already has same ip:port
     */
    public Server addServer(String serverName, String hostAddress, int port) throws Exception;

    /**
     * Remove existing {@link Server}
     *
     * @param serverName
     * @throws Exception
     *             Exception if no Server with the passed name exist or Server
     *             is started. Before removing server, it should be stopped
     */
    public void removeServer(String serverName) throws Exception;

    /**
     * Start the existing Server
     *
     * @param serverName
     *            name of the Server to be started
     * @throws Exception
     *             Exception if no Server found for given name or Server already
     *             started
     */
    public void startServer(String serverName) throws Exception;

    /**
     * Stop the Server.
     *
     * @param serverName
     *            name of the Server to be stopped
     * @throws Exception
     *             Exception if no Server found for given name or any of the
     *             {@link Association} within Server still started. All the
     *             Association's must be stopped before stopping Server
     */
    public void stopServer(String serverName) throws Exception;

    /**
     * Get the list of Servers configured
     *
     * @return
```

```
     */
    public List<Server> getServers();

    /**
     * Add server Association.
     *
     * @param peerAddress
     *            the peer IP address that this association will accept
     *            connection from
     * @param peerPort
     *            the peer port that this association will accept connection
     *            from
     * @param serverName
     *            the Server that this association belongs to
     * @param assocName
     *            unique name of Association
     * @return
     * @throws Exception
     */
    public Association addServerAssociation(String peerAddress, int peerPort, String serverName, String assocNam

    /**
     * Add Association
     *
     * @param hostAddress
     * @param hostPort
     * @param peerAddress
     * @param peerPort
     * @param assocName
     * @return
     * @throws Exception
     */
    public Association addAssociation(String hostAddress, int hostPort, String peerAddress, int peerPort, String asso

    /**
     * Remove existing Association. Association should be stopped before
     * removing
     *
     * @param assocName
     * @throws Exception
     */
    public void removeAssociation(String assocName) throws Exception;

    /**
```

```
 * Get existing Association for passed name
 *
 * @param assocName
 * @return
 * @throws Exception
 */
public Association getAssociation(String assocName) throws Exception;

/**
 * Get configured Association map with name as key and Association instance
 * as value
 *
 * @return
 */
public Map<String, Association> getAssociations();

/**
 * Start the existing Association
 *
 * @param assocName
 * @throws Exception
 */
public void startAssociation(String assocName) throws Exception;

/**
 * Stop the existing Association
 *
 * @param assocName
 * @throws Exception
 */
public void stopAssociation(String assocName) throws Exception;

/**
 * Get connection delay. If the client side {@link Association} dies due to
 * network failure or any other reason, it should attempt to reconnect after
 * connectDelay interval
 *
 * @return
 */
public int getConnectDelay();

/**
 * Set the connection delay for client side {@link Association}
 *
```

```
     * @param connectDelay
     */
    public void setConnectDelay(int connectDelay);

    /**
     * Number of threads configured to callback {@link AssociationListener}
     * methods.
     *
     * @return
     */
    public int getWorkerThreads();

    /**
     * Number of threads configured for callback {@link AssociationListener}
     *
     * @param workerThreads
     */
    public void setWorkerThreads(int workerThreads);

    /**
     * If set as single thread, number of workers thread set has no effect and
     * entire protocol stack runs on single thread
     *
     * @return
     */
    public boolean isSingleThread();

    /**
     * Set protocol stack as single thread
     *
     * @param singleThread
     */
    public void setSingleThread(boolean singleThread);
}
```

Management API is divided into two sections 1) managing the resources and 2) configuring management

4.1.1. API's to manage resources

`public Association addAssociation(String hostAddress, int hostPort, String peerAddress, int peerPort, String assocName)`

Add's a new client side association to the management. Association when started will create underlying SCTP socket that will bind to hostAddress:hostPort and tries to connect to peerAddress:peerPort. Each association is identified by unique name. The connection attempt be will made after every `connectDelay` milliseconds till the connection is successfully created.

Appropriate Exception's are thrown if other association with same name already exist or if other association is already bound to same hostAddress:hostPort or other association is already configured to connect to same peerAddress:peerPort.

`public Association addServerAssociation(String peerAddress, int peerPort, String serverName, String assocName)`

Add's a new server side association to the management. A server by name `serverName` should already have been added to the management before adding server side association. Only Association from peerAddress:peerPort will be accepted by underlying server socket. If connection request is coming from any other ip:port combination it's gracefully closed and error message is logged. If connect request comes for configured peerAddress:peerPort, but underlying association is not started, it's gracefully closed and error message is logged.

Each association is identified by unique name. The connection attempt be will made after every `connectDelay` milliseconds till the connection is successfully created.

Appropriate Exception's are thrown if other association with same name already exist or if other association is already configured to receive connection request from same peerAddress:peerPort.

`public Server addServer(String serverName, String hostAddress, int port)`

Add's a new server to the management. Server will be bound to hostAddress:port when started.

Each server is identified by unique name.

Appropriate Exception's are thrown if other server with same name already exist or if other server is already configured to bind to same hostAddress:port

`public void startAssociation(String assocName)`

Start's the association with name `assocName`. `AssociationListener` should be set before starting this association

Appropriate Exception's are thrown if there is no association with given name or if association with given name is found but is already started.

`public void startServer(String serverName)`

Start's the server with name `serverName`.

Appropriate Exception is thrown if there is no server with given name or if server with given name is found but is already started.

`public void stopAssociation(String assocName)`
: stop's the association with name `assocName`. The underlying socket is closed.

  Appropriate Exception is thrown if there is no association with given name.

`public void stopServer(String serverName)`
: stop's the server with name `serverName`.

  Appropriate Exception is thrown if there is no server with given name. Throws exception if the server is found for given name but there are association's for this server which are still in "started" state.

`public void removeAssociation(String assocName)`
: Removes the association with name `assocName`.

  Appropriate Exception is thrown if there is no association with given name. Throws exception if association is found with given name but is started.

`public void removeServer(String serverName)`
: Removes the server with name `serverName`.

  Appropriate Exception is thrown if there is no server with given name. Throws exception if server is found with given name but is started.

`public Association getAssociation(String assocName)`
: Returns the association with name `assocName`.

  Appropriate Exception is thrown if there is no Association with given name.

`public Map<String, Association> getAssociations()`
: Returns the unmodifiable Map of association. Key is association name and value is association instance

`public List<Server> getServers()`
: Returns the unmodifiable lis of servers.

4.1.2. Configuration

`setPersistDir`
: Management when started looks for file XXX_sctp.xml containing serialized state of underlying association and server. Set the directory path to direct Management to look at specified directory for underlying serialized file.

  If directory path is not set, Management searches for system property `sctp.persist.dir` to get the path for directory. Even if `sctp.persist.dir` system property is not set, Management will look at System set property `user.dir`

`setConnectDelay`
: Time in milli seconds that underlying SCTP socket will wait before attempting to connect to peer. This is only applivable for clien side sockets.

`setWorkerThreads`
: Number of threads to callback the AssociationListener. Its assured that packets with same SLS will always be queued in same Thread for callback to make sure order is maintained.

`setSingleThread`
: Only single thread will be used to callback the AssociationListener. If this is set to true, setting number of threads by calling setWorkerThreads has no effect.

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.

The Application using Mobicents 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;

/**
 * <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 {

    /**
     * Each association has unique name
     *
     * @return name of association
     */
    public String getName();

    /**
     * If this association is started by management
     *
     * @return
```

```
	 */
	public boolean isStarted();

	/**
	 * 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();

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

	/**
	 * 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);

}
```

Example

This chapter tours around the core constructs of Mobicents SCTP Library with simple examples to let you get started quickly. You will be able to write a client and a server on top of Mobicents SCTP Library right away when you are at the end of this chapter.

6.1. Before Getting Started

The minimum requirements to run the examples which are introduced in this chapter are only two; the latest version of Mobicents SCTP Library and JDK 1.7 or above with SCTP support. At time of writing this guide linux kernel has native support for SCTP (lksctp) and also Solaris includes SCTP.

6.2. Initiating Management

The primitive step in uisng Mobicents SCTP Library is to create instance of `Management` class and set appropriate parameters.

```
private static final String SERVER_NAME = "testserver";
private static final String SERVER_HOST = "127.0.0.1";
private static final int SERVER_PORT = 2345;

private static final String SERVER_ASSOCIATION_NAME = "serverAsscoiation";
private static final String CLIENT_ASSOCIATION_NAME = "clientAsscoiation";

private static final String CLIENT_HOST = "127.0.0.1";
private static final int CLIENT_PORT = 2346;
...
....
....

Management management = new ManagementImpl("SCTPTest");(1)
management.setConnectDelay(10000);// Try connecting every 10 secs(2)
management.setSingleThread(true);(3)
management.start();
```

(1) Crate new instance of `ManagementImpl` and setting the management name. The management will search for SCTPTest_SCTP.xml file to load the previously configured Serever's or Association's. If file is not found it will create one.

(2) connectDelay is only useful for Associations acting as client side trying to connect to peer. The value specified is time in milliseconds the unedrlying socket will try to connect to peer if the existing connection is broken or even if its fresh connection attempt.

(3) Setting management to single thread. All the callback's (irrespective of streamNumber of packet) to the listener will be via single thread. However there is one dedicated thread only for I/O.

6.3. Adding Server and server Association

Once the Managment is setup, application can create `Server` and add server `Association`

```
        Server server = this.management.addServer(SERVER_NAME, SERVER_HOST, SERVER_PORT);
    Association serverAssociation = this.management.addServerAssociation(CLIENT_HOST, CLIENT_PORT, SERVER_NAME, SERVER_ASSOCIATION_NAME); (1)

 serverAssociation.setAssociationListener(new ServerAssociationListener()); (2)

 this.management.startAssociation(SERVER_ASSOCIATION_NAME);
 this.management.startServer(SERVER_NAME);
```

(1) Add the server and server association. Multiple server's can be added to each management and each server can have multiple server association's

(2) The instance of `AssociationListener` should be registered with newly created Associaton before starting it. There is no dependency on order of starting server and server association.

Below is example of class implementing `AssociationListener`

```
class ServerAssociationListener implements AssociationListener {

 private final byte[] SERVER_MESSAGE = "Server says Hi".getBytes();

 /*
  * (non-Javadoc)
  *
  * @see
  * org.mobicents.protocols.sctp.AssociationListener#onCommunicationUp
  * (org.mobicents.protocols.sctp.Association)
  */
 @Override
 public void onCommunicationUp(Association association) {
  System.out.println(this + " onCommunicationUp");

  serverAssocUp = true;

            PayloadData payloadData = new PayloadData(SERVER_MESSAGE.length,
SERVER_MESSAGE, true, false, 3, 1);

  try {
   association.send(payloadData);
  } catch (Exception e) {
   e.printStackTrace();
  }
 }

 /*
  * (non-Javadoc)
  *
  * @see
  * org.mobicents.protocols.sctp.AssociationListener#onCommunicationShutdown
  * (org.mobicents.protocols.sctp.Association)
  */
 @Override
 public void onCommunicationShutdown(Association association) {
  System.out.println(this + " onCommunicationShutdown");
  serverAssocDown = true;
 }

 /*
  * (non-Javadoc)
  *
  * @see
  * org.mobicents.protocols.sctp.AssociationListener#onCommunicationLost
  * (org.mobicents.protocols.sctp.Association)
  */
 @Override
 public void onCommunicationLost(Association association) {
  System.out.println(this + " onCommunicationLost");
 }

 /*
  * (non-Javadoc)
  *
	 * @see
	 * org.mobicents.protocols.sctp.AssociationListener#onCommunicationRestart
	 * (org.mobicents.protocols.sctp.Association)
	 */
	@Override
	public void onCommunicationRestart(Association association) {
		System.out.println(this + " onCommunicationRestart");
	}

	/*
	 * (non-Javadoc)
	 *
	 * @see
	 * org.mobicents.protocols.sctp.AssociationListener#onPayload(org.mobicents
	 * .protocols.sctp.Association,
	 * org.mobicents.protocols.sctp.PayloadData)
	 */
	@Override
	public void onPayload(Association association, PayloadData payloadData) {
		System.out.println(this + " onPayload");

		serverMessage = new byte[payloadData.getDataLength()];
	System.arraycopy(payloadData.getData(), 0, serverMessage, 0, payloadData.getDataLength());

		System.out.println(this + "received " + new String(serverMessage));
	}

}
```

6.4. Adding Association

Once the Managment is setup, application can create client side `Association`.

```
		Association clientAssociation = this.management.addAssociation(CLIENT_HOST,
 CLIENT_PORT, SERVER_HOST, SERVER_PORT, CLIENT_ASSOCIATION_NAME);
	clientAssociation.setAssociationListener(new ClientAssociationListenerImpl());
	this.management.startAssociation(CLIENT_ASSOCIATION_NAME);
```

Appendix A. Revision History

Revision History		
Revision 1.0	Fri Nov 25 2011	AmitBhayani
Creation of the Mobicents SCTP Library User Guide.		

Index