Configuring as a Standalone library

Table of Contents

If you intend to use the Stack as a standalone library without using JBoss Application Server or JSLEE RAs, then you must manually build each of the protocols, configure them individually, and bind them together.

Building SCTP

This is an example of a typical SCTP startup procedure, in one line of code, using automatic configuration file detection:

```
        org.mobicents.protocols.api.Management sctpManagement = null;
        sctpManagement = org.mobicents.protocols.sctp.ManagementImpl("Client");
        this.sctpManagement.setSingleThread(true);
        this.sctpManagement.setConnectDelay(10000);
        this.sctpManagement.start();
```

How did jSS7 know where the configuration file was located and which one to load?

When `this.sctpManagement.start()` is called, jSS7 searches for a file named *Client_sctp.xml* in the directory path set by user by calling `this.sctpManagement.setPersistDir("<your directory path>")`. For example in case of linux you can pass something like `this.sctpManagement.setPersistDir("/home/abhayani/workarea/mobicents/git/jss7/master/map/load/client")`. 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`.

Once you know SCTP layer is configured and started, next step is add the Association and/or Server depending on whether this setup will be acting as client or server or both.

- For client side : `sctpManagement.addAssociation(CLIENT_IP, CLIENT_PORT, SERVER_IP, SERVER_PORT, CLIENT_ASSOCIATION_NAME, ipChannelType, null);`
- For server side : `sctpManagement.addServerAssociation(CLIENT_IP, CLIENT_PORT, SERVER_NAME, SERVER_ASSOCIATION_NAME, ipChannelType);`

Before adding server side association the server should also be defined and started as below:

```
        sctpManagement.addServer(SERVER_NAME, SERVER_IP, SERVER_PORT, ipChannelType,
null);
        sctpManagement.addServerAssociation(CLIENT_IP, CLIENT_PORT, SERVER_NAME,
SERVER_ASSOCIATION_NAME, ipChannelType);
        sctpManagement.startServer(SERVER_NAME);
```

You should never start the Association programatically. Association will be started automatically when layer above M3UA's ASP is started.

This completes the SCTP configuration and start-up.

Building M3UA

Configuring the M3UA layer is similar to the steps followed for SCTP.

```
                org.mobicents.protocols.ss7.m3ua.impl clientM3UAMgmt = null;
                this.clientM3UAMgmt = new M3UAManagement("Client");
                this.clientM3UAMgmt.setTransportManagement(this.sctpManagement);
                this.clientM3UAMgmt.start();
```

For M3UA, it should know which underlying SCTP layer to use `this.clientM3UAMgmt.setTransportManagement(this.sctpManagement);`.

Once M3UA is configured and started, next step is to add the As, Asp and routing rules for M3UA. These depends on whether stack acts as Application Server side or Signaling Gateway side or just peer-to-peer (IPSP) client/server side. Below is an example of IPSP peer acting as client.

```
        RoutingContext rc = factory.createRoutingContext(new long[] { 100l });
                TrafficModeType trafficModeType =
factory.createTrafficModeType(TrafficModeType.Loadshare);
                this.clientM3UAMgmt.createAs("AS1", Functionality.AS, ExchangeType.SE,
IPSPType.CLIENT, rc, trafficModeType, null);
                // Step 2 : Create ASP
                this.clientM3UAMgmt.createAspFactory("ASP1", CLIENT_ASSOCIATION_NAME);
                // Step3 : Assign ASP to AS
                Asp asp = this.clientM3UAMgmt.assignAspToAs("AS1", "ASP1");
                // Step 4: Add Route. Remote point code is 2
                clientM3UAMgmt.addRoute(SERVET_SPC, -1, -1, "AS1");
```

This completes the M3UA configuration and start-up. Once M3UA is configured depending on whether you are trying to build voice application that depends on ISUP or advanced network features such as those offered by supplementary services that depends on MAP, you would configure ISUP or SCCP

Building SCCP

Configuring the SCCP layer follows exactly same architecture of persisting configuration in xml file.

```
        org.mobicents.protocols.ss7.sccp.SccpStack sccpStack = null;
               this.sccpStack = new SccpStackImpl("MapLoadClientSccpStack");
               this.sccpStack.setMtp3UserPart(1, this.clientM3UAMgmt);
               this.sccpStack.start();
```

Before starting SCCP stack all it needs to know is underlying MTP3 layer. Above sections explained building SCTP and M3UA, however if you are using Dialogic boards or dahdi based boards (Diguim/Sangoma), you need to build and configure respective MTP3 layers depending on

hardware used and set those in SCCP Stack `this.sccpStack.setMtp3UserPart(1, this.clientM3UAMgmt)`.

One of the best features of jSS7 is it supports multiple MTP3 layers and hence you can have combination of many MTP3 layers (each of different or same type like M3UA, Dialogic and Dahid; all used at same time).

Once SCCP stack is started, it should be configured for local and remote signaling point-code, network indicator, remote sub system number and routing rules.

```
                RemoteSignalingPointCode rspc = new
RemoteSignalingPointCode(SERVET_SPC, 0, 0);
                RemoteSubSystem rss = new RemoteSubSystem(SERVET_SPC, SSN, 0, false);
                this.sccpStack.getSccpResource().addRemoteSpc(0, rspc);
                this.sccpStack.getSccpResource().addRemoteSsn(0, rss);
                Mtp3ServiceAccessPoint sap = new Mtp3ServiceAccessPoint(1, CLIENT_SPC,
NETWORK_INDICATOR);
                Mtp3Destination dest = new Mtp3Destination(SERVET_SPC, SERVET_SPC, 0,
255, 255);
                this.sccpStack.getRouter().addMtp3ServiceAccessPoint(1, sap);
                this.sccpStack.getRouter().addMtp3Destination(1, 1, dest);
```

Once SCCP is configured and started, next step it to build TCAP layer

Building TCAP

There is no configuration to persist in case of TCAP.

```
                org.mobicents.protocols.ss7.tcap.api tcapStack = null;
                this.tcapStack = new TCAPStackImpl(this.sccpStack.getSccpProvider(),
SSN);
                this.tcapStack.setDialogIdleTimeout(60000);
                this.tcapStack.setInvokeTimeout(30000);
                this.tcapStack.setMaxDialogs(2000);
                this.tcapStack.start();
```

Configuring TCAP is probably very simple as config reamins same irrespective of whether its used on client side or server side.

Building MAP

There is no configuration to persist in case of MAP; however MAP stack can take TCAPProvider from TCAPStack which is already configured for specific SSN as shown below:

```
                this.mapStack = new MAPStackImpl(this.tcapStack.getProvider());
```

Or it can also directly take SccpProvider and pass SSN to MAP Stack as shown below. In this case MAPStack itself creates the TCAPStack and leverages TCAPProvider:

```
                this.mapStack = new MAPStackImpl(this.sccpStack.getSccpProvider(),
SSN);
```

Before MAPStack can be started, the Application interested in particualr MAP Service should register it-self as listener and activate that service:

```
        this.mapProvider = this.mapStack.getMAPProvider();
                this.mapProvider.addMAPDialogListener(this);

this.mapProvider.getMAPServiceSupplementary().addMAPServiceListener(this);
                this.mapProvider.getMAPServiceSupplementary().acivate();
                this.mapStack.start();
```

Below is how the Application code looks like:

```
         public class Client extends MAPDialogListener,
MAPServiceSupplementaryListener  {
                    //Implemet all MAPDialogListener methods here


                    //Implement all MAPServiceSupplementaryListener methods here
            }
```

Common Code

All above snippet of code refers to below defined constants:

```
        // MTP Details
        protected final int CLIENT_SPC = 1;
        protected final int SERVET_SPC = 2;
        protected final int NETWORK_INDICATOR = 2;
        protected final int SERVICE_INIDCATOR = 3; //SCCP
        protected final int SSN = 8;

        protected final String CLIENT_IP = "127.0.0.1";
        protected final int CLIENT_PORT = 2345;

        protected final String SERVER_IP = "127.0.0.1";
        protected final int SERVER_PORT = 3434;

        protected final int ROUTING_CONTEXT = 100;

        protected final String SERVER_ASSOCIATION_NAME = "serverAsscoiation";
        protected final String CLIENT_ASSOCIATION_NAME = "clientAsscoiation";

        protected final String SERVER_NAME = "testserver";

.....
.....
```

Once you have completed development of your application, next thing is setting the classpath, compiling and starting application. You must set the classpath to point to restcomm-jss7-X.Y.Z/ss7/restcomm-ss7-service/lib. It has all the libraries needed to compile and start your application. Don’t forget to include your compiled Application class file in classpath before starting the Application.