Configuring the Media Server

Table of Contents

System clock and scheduler

Scheduler is the core component of Media Server. It is responsible for organizing all tasks executed by the media server and executing them. Current scheduler implementation is a time wheel like implementation. All tasks are executed in 20ms cycle which is best optimized for audio.

Clock is the class that implements basic clock settings, providing current time and translations needed between time unit and time.

Configuration changes are required only in case you want to develop your own version of scheduler/clock.

Example 1. System Clock and Scheduler

```
<!-- System clock -->
<bean name="SystemClock" class=
"org.mobicents.media.server.scheduler.DefaultClock"/>

<!-- EDF scheduler -->
<bean name="Scheduler" class="org.mobicents.media.server.scheduler.Scheduler">
<property name="clock"><inject bean="SystemClock"/></property>
</bean>
```

UDP Manager

UDP manager is responsible for receiving all the data that comes through UDP channels to the Media Server, including RTP packets and control protocol packets, like MGCP.

Example 2. UDP Manager

```
<!-- Network interfaces definition -->
<bean name="localhost" class="org.mobicents.media.server.io.network.UdpManager">
<constructor>
<parameter><inject bean="Scheduler"/></parameter>
</constructor>
<property name="bindAddress">127.0.0.1</property>
<property name="localBindAddress">127.0.0.1</property>
<property name="localNetwork">127.0.0.1</property>
<property name="localSubnet">255.255.255.255</property>
<property name="useSbc">false</property>
<property name="rtpTimeout">0</property>
</bean>
```

bindAddress and localBindAddress.

Current release of MMS allows 2 network interfaces, a local and a remote interface. By default all RTP channels are opened on bindAddress, while MGCP control interface starts on localBindAddress. Using both interfaces is useful in the following scenarios:

When you want to separate control traffic from data traffic.

When you want an SBC like border media server, in this case, the remote interface will send/ receive data from the UA. The local interface will be used for inter-server traffic and control traffic (MGCP).

To establish RTP connection on local interface, you should add TypeOfNetwork.Local local connection option to CRCX (create connection) of MGCPrequest.

Any RTP connection that will be opened on remote interface will have jitter buffer enabled , while any RTP connection that will be opened on local interface will have jitter buffer disabled.

LocalNetwork, localSubnet and useSbc.

Media server can work in 2 modes : SBC mode and standard mode.

When useSbc is set to false, MMS will establish a channel based on address/port pair it receives from SDP. However, RTP packet is used within a NAT context, data sent in SDP is the original data and not the natted IP address/port (this is often the scenario). Furthermore, when useSbc is set to false, data will be sent to invalid address and also will not be accepted since MMS does not know the correct address of UA.

In order to resolve the above mentioned issue, you should set useSbc to true. As a result, MMS will wait for first RTP packet; learn remote IP address port and then will send data based on learned address and not on SDP data. That said, inter-server traffic will not work since both sides will be waiting for first packet. To resolve this, you should set local network and local subnet to range that will include all your media servers. If the media server detects that the IP address and port in SDP are within the local IP network range, it will start sending packets immediately and not wait. This is similar to when useSbc is set to false.

RtpTimeout

Most SIP UA do not support any type of keepalive between 200 OK and BYE response/ request. Therefore, if the network goes down while a call is established, the call may hang forever. That is why RTP traffic should always be sent (the exception to this rule is during recvonly and inactive modes). With the this in mind, MMS enables RTP timeout parameter.

When RtpTimeout is set to greater than 0, MMS will monitor RTP traffic and if it finds period equal or greater then the RTP timeout (in seconds) it will delete connection and notify controlling server that connection was removed (by sending DLCX MGCP command). Consequently, any border server will receive a reliable notification that the call is still alive even when the communication network is no longer available.

Note that while enabling RTP timeout, it is recommended that you do not set the mode to inactive/ sendonly when you expect to receive data (after 180 or 200 OK) depending on your application

Example 3. Call Flow

```
Similar call flow may be like this

UA ----> INVITE ----> Control Server
Control Server ----> CRCX with mode inactive ----> Media Server
Control Server ----> INVITE ----> inside network or to other side
Inside network or other side ----> 183 with SDP ----> Control Server
Control Server ---> MDCX with mode sendonly ---> Media Server
Control Server ---> 183 with SDP ----> UA
Inside network or other side ----> 200 ----> Control Server
Control Server ---> MDCX with mode sendrecv ---> Media Server
Control Server ---> 200 ----> UA
```

In case of 180 or 183 without SDP response , intermediate MDCX is not required.

SS7 Channels Manager

Current release of Media Server supports SS7 channels(DS0) makes it possible to communicate with old PSTN networks.

To allow SS7 B channels to be used, you must install and configure your card properly and enable SS7 manager, dahdi native driver and configure DS0 channels on Media Server. By default SS7 is disabled.

Below you can see the SS7 manager example configuration. Changes are required only when you want to implement your own version of the manager.

Example 4. Channel Manager

```
    <!-- SS7 interface definition -->

  <!--
  <bean name="SS7" class="org.mobicents.media.server.io.ss7.SS7>
  <constructor>
  <parameter><inject bean="Scheduler"/></parameter>
  </constructor>
  </bean>
  -->
```

To compile and create dahdi native library you must get a Media Server code from

https://github.com/RestComm/mediaserver by using a GIT

Once you have the source, you need to build it using Maven from a terminal console as shown below:

```
mvn clean install -p dahdilinux
```

Under *bootstrap/target/mms-server* you will have a newly compiled version of media server. There will be a native dahdi driver included in the sub-directories

Known Issues

Currently only dahdi based cards are supported Only linux OS is supported for SS7

You must compile Media Server on the production server on which it will be used. This must be done for each server that you plan to put in service.

For help on configuring DS0 channels please see Endpoints configuration section .

Channel Manager

Channel manager is responsible for creating all data channels on media server(RTP data channel , local channel and SS7 channel).

Below you can see default configuration. By default, SS7 is disabled and should be enabled only in case you want to use DS0 channels. Apart from SS7, no other configuration changes are required.

Example 5. Channel Manager

```
<!-- Channels manager definition -->
<bean name="channelsManager"
class="org.mobicents.media.server.impl.rtp.ChannelsManager">
<constructor>
<parameter><inject bean="localhost"/></parameter>
</constructor>
<property name="scheduler"><inject bean="Scheduler" /></property>
<property name="jitterBufferSize">50</property>
<!--<property name="SS7Manager" > <inject bean="SS7" /></property>-->
</bean>
```

Jitter buffer size

Jitter buffer size parameter allows you to set the jitter buffer in milliseconds. Please see the Channel Manager for details about configuring jitterBufferSize.

DSP Factory

DSP factory allows transcoding between different codecs. Currently media server comes with 5 codecs : G711 A/U, Linear PCM Raw, GSM and G.729. ILBC codec is currently being worked on and will be released in future iterations of the Media Server.

By default, only 3 are enabled : G711A/U and linear.

Example 6. Default Codecs

```
     <!-- Signal processor factory -->
        <bean name="DSP"
class="org.mobicents.media.server.component.DspFactoryImpl">
        <property name="codecs">
        <list value-type="java.lang.String">
        <value>org.mobicents.media.server.impl.dsp.audio.l16.Encoder</value>
        <value>org.mobicents.media.server.impl.dsp.audio.l16.Decoder</value>
        <value>org.mobicents.media.server.impl.dsp.audio.g711.alaw.Encoder</value>
        <value>org.mobicents.media.server.impl.dsp.audio.g711.alaw.Decoder</value>
        <value>org.mobicents.media.server.impl.dsp.audio.g711.ulaw.Encoder</value>
        <value>org.mobicents.media.server.impl.dsp.audio.g711.ulaw.Decoder</value>
        </list>
        </property>
        </bean>
```

To enable G729 codec you should add the following values :

```
      org.mobicents.media.server.impl.dsp.audio.g729.Encoder
      org.mobicents.media.server.impl.dsp.audio. g729.Decoder
```

G.729 usage

Please note that a valid license is required to use G.729 , therefore you should purchase a license prior to enabling this codec.

To enable GSM codec you should add the following values :

```
org.mobicents.media.server.impl.dsp.audio.gsm.Encoder
org.mobicents.media.server.impl.dsp.audio.gsm.Decoder
```

If you decide to use a single codec for encoding or decoding data, you should leave one RAW or 2 Raw pair. This is useful only in case of a one way activity.

Use of L16 codec

L16 codec is useful only in server to server communication where you have enough network bandwidth. It is not recommended to allow L16 codec for UA – server connections, this can lead to degradation of the signal quality due to increased jitter and packet loss.

Resource Pool

In the current Media Server release, global pool of resources is used to decrease garbage collection and allow faster resources allocation. Any resource may be used by any endpoint. For example, RTP connection 1 may be for endpoint 1 and then released and reused by endpoint 2.

You can see default configuration below :

Example 7. Default Codecs

```
<!-- Resources pool definition -->
  <bean name="resourcesPool" class="org.mobicents.media.core.ResourcesPool">
  <constructor>
  <parameter><inject bean="Scheduler"/></parameter>
  <parameter><inject bean="channelsManager"/></parameter>
  <parameter><inject bean="DSP"/></parameter>
  </constructor>
  <property name="defaultPlayers">5</property>
  <property name="defaultRecorders">5</property>
  <property name="defaultDtmfDetectors">5</property>
  <property name="defaultDtmfGenerators">0</property>
  <property name="defaultSignalDetectors">0</property>
  <property name="defaultSignalGenerators">0</property>
  <property name="defaultLocalConnections">10</property>
  <property name="defaultRemoteConnections">10</property>
  <property name="dtmfDetectorDbi">-35</property>
  </bean>
```

As can be seen above, the default pool size is configured for player(audio player), recorder (audio recorder), DTMF generator (for generation out of band DTMF tones through sl MGCP package), DTMF detector(to detect both inband/out of band tones), signal detector(used to detect ss7 tones) and signal generator(used to generate ss7 tones).

Signal detector and signal generator are currently used only for connectivity tests for DS0 channel (COT isup signal), CO1, CO2, CT(Continuity transport) and Loopback test modes are supported by the Media Server.

For more information please see [_msep_ms_event_packages]

Local connections and remote connections pools are also configured here.

When the specified resource type is not available

Please note that a resource will be automatically allocated if the specified resource type is not available in the resource pool. This will require more memory allocation and in some cases may impact performance. The more resources you have preconfigured on startup in the resource pool, the more memory the media server will require on startup. Its up to you to decide the best trade-off for your setup(greater memory usage on startup vs slower response when new resources are required in runtime)

DtmfDetectorDbi

Often, audio data is mixed with DTMF inband tones. As a result, Media Server may detect false positive tones, or it may not detect tones which are sent. By setting DTMF detector dbi parameter, you can optimize tone detection for your server. If you have problems with inband DTMF detection, you can fine-tune this parameter. However, default value has been tested and found to be the most appropriate. For best results it, is recommended that you use inband tones only in SS7/IP mixed network. Out-of-band tones are recommended for IP only networks.

MGCP Controller

The controller is the main component that allows MGCP control protocol. Enabling MGCP is always required as the jsr-309 implementation is based on the current MGCP implementation.

Example 8. MGCP Controller

```
        <!-- MGCP Controller definition -->
        <bean name="MGCP" class
="org.mobicents.media.server.mgcp.controller.Controller">
        <property name="udpInterface"><inject bean="localhost"/></property>
        <property name="port">2427</property>
        <property name="scheduler"><inject bean="Scheduler"/></property>
        <property name="server"><inject bean="MediaServer"/></property>
        <property name="configuration">mgcp-conf.xml</property>
        <property name="poolSize">25</property>
        </bean>
```

Port Parameter

UDP port is used for MGCP traffic. By default, it is set to 2427. However you can change it to whatever you want to. Note that you will have to change your control server configuration/code in case you decide to use a number different from the default.

Configuration parameter

Configuration parameter points to the xml file located in [path]_ conf_ directory which configures MGCP packages, signals and packages mapping to endpoints.

Pool Size

Pool Size is the size of MGCP requests that Media Server will handle concurrently. The current MGCP offers better performance and ensures that new elements are allocated when there isn't enough in the pool (most requests are executed under 4ms). That said, RQNT can take up 20ms.

Media Server

Media server component is the core of the software. Its job is to start all the elements and stop them when the Media Server is stopped. It is recommended that you only change the heartBeatTime parameter if required and leave the rest as default.

Example 9. MGCP Controller

```
            <!-- Media Server -->
            <bean name="MediaServer" class="org.mobicents.media.core.Server">
            <property name="clock"><inject bean="SystemClock"/></property>
            <property name="scheduler"><inject bean="Scheduler"/></property>
            <property name="udpManager"><inject bean="localhost"/></property>
            <property name="resourcesPool"><inject bean=
"resourcesPool"/></property>
            <property name="heartBeatTime">0</property>
            <incallback method="addInstaller"/>
            <uncallback method="removeInstaller"/>
            </bean>
```

Heartbeat Time

When you need to debug Media Server, you can set heartBeatTime to a specific value(in seconds). In this case, on each heartBeatTime period, MMS will write one raw to log and will notify that it is alive. This is useful when you suspect communication problems with the Media Server. It will help you get closer to the source of the problem. Enabling hearbeat will let you know if the server is up and running.

Endpoints

Endpoints configuration allows you to configure all MGCP endpoints groups you want to use with the Media Server instance. For all endpoints types, you can configure the initial size. This value will determine the number of endpoints that are preloaded on startup. If all available endpoints are

used and a request for additional endpoint is received, Media Server will allocate a new endpoint and store it in a resource pool. The only exception to this rule is DS0 endpoint type. The DS0 endpoint can not be increased as it is directly related to the number of channels available on an E1/T1 card.

NamePattern is the name to use for all MGCP requests. Note that an integer value will be appended to the name used. For example, a NamePattern called "mobicents/aap" will be created and accessed as "mobicents/aap/$" (where $ is an interger).

Class is the class of installer. Currently 2 types of installers are available : *VirtualEndpointInstaller* which allows you to install most endpoints and *VirtualSS7EndpointInstaller* which is used for DS0 endpoints.

Endpoint class – defines the class which implements endpoints.

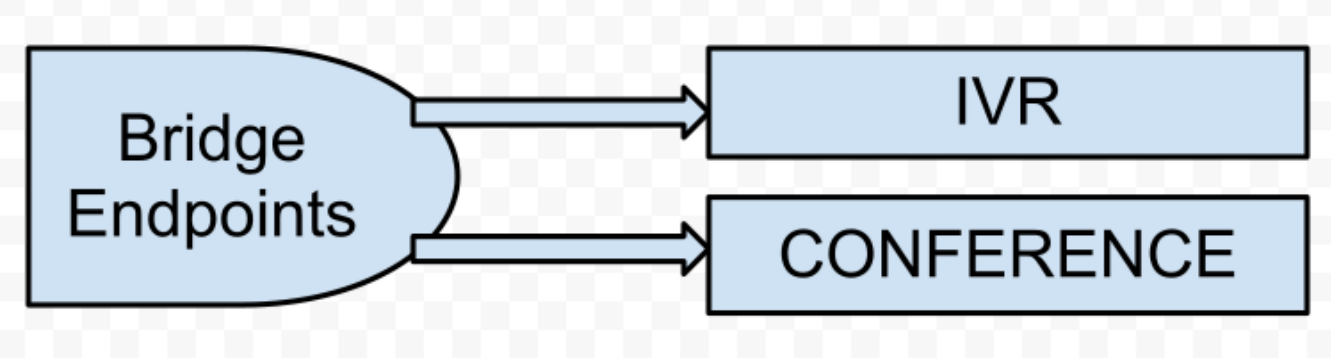
Currently, the following endpoint classes are available :

Table 1. Endpoint Classes

Endpoint Class	Available Resources	Connection Types	Types
org.mobicents.media.core.endpoints.impl.AnnouncementEndpoint	Player	Local+RTP	Mixer
org.mobicents.media.core.endpoints.impl.IvrEndpoint	Player,Recorder,Dtmf Detector,Dtmf Generator	Local+RTP	Mixer
org.mobicents.media.core.endpoints.impl.ConferenceEndpoint	None	Local+RTP	Mixer
org.mobicents.media.core.endpoints.impl.BridgeEndpoint	None	Local+RTP	Splitter
org.mobicents.media.core.endpoints.impl.PacketRelayEndpoint	None	RTP	Mixer
org.mobicents.media.core.endpoints.impl.Ds0Endpoint	Signal Detector,Signal Generator	Local+RTP	Splitter

Mixer Type means all connections and resources data will be mixed together. This implies that it will still be available even without the appropriate setting mode.

Bridge endpoint is not the same as the MGCP standard endpoint type. This means that there are 2 resources groups. The first group uses local connections whereas the second group uses RTP connections. This implies that non RTP connections can communicate with RTP endpoints and none local connections can still access local endpoints. The Bridge endpoint type is useful in the scenarios shown below:



IVR and Conference endpoints will be connected through Local connection pair to Bridge Endpoint (which is recommended). There will be no cross over of IVR Resources (player , recorder , etc) to conference. The same applies to any conference traffic, it will not cross over to IVR. This is useful for recording RTP data for specific groups of users.

DS0 endpoint is a type Splitter. All connections, signal detector and signal generator are in group 1, while signalling channel SS7 is placed in group 2. That means that any SS7 channel data will be sent to any connection and signal detector, while data from signal generator and any connection will be sent only to SS7 channel.

Endpoint Groups

In order to configure multiple groups of endpoints of the same type per Media Server instance, you must change the name of each group.

Below you can see an example for endpoints configuration

Example 10. Endpoints Configuration

```
            <!-- Endpoints -->
<bean name="Announcement"
class="org.mobicents.media.core.endpoints.VirtualEndpointInstaller">
<property name="namePattern">mobicents/aap/</property>
<property
name="endpointClass">org.mobicents.media.core.endpoints.impl.AnnouncementEndpoint<
/
property>
<property name="initialSize">1</property>
</bean>

<bean name="IVR"
class="org.mobicents.media.core.endpoints.VirtualEndpointInstaller">
<property name="namePattern">mobicents/ivr/</property>
<property name="endpointClass">
org.mobicents.media.core.endpoints.impl.IvrEndpoint</
property>
<property name="initialSize">5</property>

</bean>
```

```
<bean name="CNF"
class="org.mobicents.media.core.endpoints.VirtualEndpointInstaller">
<property name="namePattern">mobicents/cnf/</property>
<property
name="endpointClass">org.mobicents.media.core.endpoints.impl.ConferenceEndpoint</
property>
<property name="initialSize">5</property>
</bean>

<bean name="Bridge"
class="org.mobicents.media.core.endpoints.VirtualEndpointInstaller">
<property name="namePattern">mobicents/bridge/</property>
<property
name="endpointClass">org.mobicents.media.core.endpoints.impl.BridgeEndpoint</prope
rty>
<property name="initialSize">5</property>
</bean>

<bean name="Relay"
class="org.mobicents.media.core.endpoints.VirtualEndpointInstaller">
<property name="namePattern">mobicents/relay/</property>
<property
name="endpointClass">org.mobicents.media.core.endpoints.impl.PacketRelayEndpoint</
property>
<property name="initialSize">1</property>
</bean>

<!-- DS0 Endpoints configuration sample -->
<!--<bean name="DS0"
class="org.mobicents.media.core.endpoints.VirtualSS7EndpointInstaller">
<property name="namePattern">mobicents/ds0/</property>
<property name="channelsManager"><inject bean="channelsManager"/></property>
<property
name="endpointClass">org.mobicents.media.core.endpoints.impl.Ds0Endpoint</
property>
<property name="startChannelID">125</property>
<property name="initialSize">15</property>
<property name="isALaw">true</property>
</bean>-->
```

SS7 Default

SS7 (DS0) endpoints are disabled by default. There are additional configurable parameters.

startChannelID

Start channel id is the first CIC number. For example, if you have 4E1 card and want to allocate to specific group 50 channels starting from channel 5, on E1 number 2 , you will have to set start

channel id to 35 (30 channels per E1) and initial Size to 50. Once agai, the number of DS0 will not be increased by Media Server.

Channels Manager

Channels manager is the bean mention in the section Channel Manager