IP Video Systems
Media Room Agent API

IP Video Systems Media Room Agent API

Table of Contents

List of Examples

Glossary

MediaRoom	The MediaRoom is an extension of a multiuser chat room whose participants are media objects which fulfill certain roles in the room.
MediaTimeline	The MediaTimeline is a property of the MediaRoom which includes information about the Start of the MediaRoom and the Current Time-Shifted state
MediaTrack	The MediaTrack is a grouping of media objects for control purposes which share certain common MediaTimeline and properties
Media Source	The MediaSource is a media object which fulfills the role of a source capable of generating a media stream
MediaDest	The MediaDest is a media object which fulfills the role of a sink capable of receiving a media stream
Stream	The stream represents the data that flows from a Media Source to a Media Dest
StreamingServer	A server capable of sending and receiving media streams
AppServerAgent	The agent responsible for originating requests for setup/teardown/PVR of streams
SourceStreamAgent	The agent hosted by a media source which handles requests for starting/stopping/ PVR of streams
DestStreamAgent	The agent hosted by a media dest which handles requests for setup/teardown of streams
MediaRelay	The MediaRelay is a media object which receives a stream from a media source and relays it to a media dest

Introduction

This document contains parts of the Media Room Agent API specifications pertaining to the setup and control of Media Rooms.

Chapter 1. Service Overview

This chapter provides the version information and a brief description of the Agents involved in the IPVS Mediaroom Service.

Version

Document Version = 0.0.1
Service Version = 2.24
Software Version = 3.8.x.x

MediaRoom Overview

The MediaRoom is an extension of a multiuser chat room whose participants are media objects which fulfill certain roles in the room. The goal of the MediaRoom service is to allow media to be viewed/ recorded/controlled in a chat room, thus enabling various collaboration applications This goal is realized by Agents implemented by devices which provide the MediaStream setup/control/teardown functionality Each of the devices implements a StreamingServer which ultimately receives/sends a MediaStream The Agent messages ultimately result in a command being sent to the streaming server on the device hosting the Agent This chapter provides an overview of the agents, their roles and messaging between the agents to realize the MediaRoom functionality.

Agents and Roles

This section lists the primary agents and their roles.

AppServerAgent: Hosted by a central server, responsible for handling requests to setup MediaRooms and Streams
DestStreamAgent: Hosted by each MediaDestination, responsible for handling requests to request a Stream from a source to the device on which it is co-located
SourceStreamAgent: Hosted by each MediaSource, responsible for handling requests for a starting/ controlling a Stream

Message Flow to Setup A Media Stream

This section lists a sample flow of messages to setup a Media Stream.

AppServerAgent: Receives a request from the collaboration application to add a stream
AppServerAgent: Handles the request and sends a "Setup" request to the DestStreamAgent
DestStreamAgent: Handles the "Setup" request and sends a "Start" request to the SourceStreamAgent
SourceStreamAgent: Handles the "Start" request, adds a stream in the co-located Streaming Server and sends the "StartResponse" with the URL for the stream to the DestStreamAgent
DestStreamAgent: Handles the "StartResponse" and passes the URL to the co-located StreamingServer

Message Flow to Teardown A Media Stream

This section lists a sample flow of messages to teardown a Media Stream.

AppServerAgent: Receives a request from the collaboration application to delete a stream

AppServerAgent: Handles the request and sends a "Teardown" request to the DestStreamAgent
DestStreamAgent: Handles the "Teardown" request, deletes the URL from the co-located Streaming Server and sends a "Stop" request to the SourceStreamAgent
SourceStreamAgent: Handles the "Stop" request, deletes the stream in the co-located Streaming Server and sends the "StopResponse" with the URL for the stream to the DestStreamAgent
DestStreamAgent: Handles the "StopResponse"

Message Flow to Control A Media Stream

This section lists a sample flow of messages to control(Pause/Resume) a Media Stream.

AppServerAgent: Receives a request from the collaboration application to control a stream
AppServerAgent: Handles the request, translates the control to an Update(timescale + source + offset) and sends a "Update" request to the SourceStreamAgent
SourceStreamAgent: Handles the "Update" request, deletes the previous stream and sets up the updated stream in the co-located Streaming Server and sends the "UpdateResponse" to the DestStreamAgent
DestStreamAgent: Handles the "UpdateResponse"

Message Flow to Update the State Stream

This section lists a sample flow of messages in case there is a State update for Media Stream.

SourceStreamAgent: Generates a SourceEvent message and sends to the DestStreamAgent
DestStreamAgent: Handles the SourceEvent, generates a StreamStatusUpdate event and sends to the AppServerAgent
AppServerAgent: Handles the StreamStatusUpdate and notifies the collaboration application

Chapter 2. Agent Message Overview

All API calls are transacted as extensions of standard "XMPP Message" packets.

All IPVS agent message payloads will use the namespace "com.ipvs.agentmessage"

Generic Agent Message API

The messaging between the agentts is realized as a set of XML messages over an XMPP message bus. These messages constitute the IPVS Agent API. Agents will need to implement the relevant sub-set of this API based on their intended MediaRoom Role.

Example 2.1. Message

```
  <message id="12rLk-221" to="tx1@localhost/tx1"
 from="managementserver@localhost/managementserver">
    <x>
      <AgentMessage>
        <agentmessageheader>
          <fromagent>com.ipvs.mediaroomservice.impl.AppServerAgent</
fromagent>
          <agentJID>tx1@localhost/tx1</agentJID>
          <agent>com.ipvs.client.MRClientAgent</agent>
          <agentmessagetype>
        <!-- Message Type goes here: Request/Response/AgentEvent -->
          </agentmessagetype>
        </agentmessageheader>
        <agentmessagedata>
        <!-- Message Data goes here -->
        </agentmessagedata>
     </AgentMessage>
    <x>
  </message>
```

Generic Agent Message Data For Request Type Agent Messages

Example 2.2. Request

```
<Request>
  <header>
      <servicename>xmpp</servicename>
      <requestname>
     <!-- Request name goes here -->
     SetServiceDomainRequest
   </requestname>
      <loglevel>0</loglevel>
      <clientdata>RequestToken2</clientdata>
  </header>
  <data>
     <!-- Request data goes here -->
     <SetServiceDomainRequestData>
             <!-- SetServiceDomainRequestData payload goes here -->
     </SetServiceDomainRequestData>
  </data>
</Request>
```

Generic Agent Message Data For Response Type Agent Messages

Example 2.3. Response

```
 <Response serviceVer="2.25">
            <header>
              <servicename>systemservice</servicename>
              <requestname>
         <!-- Request name goes here -->
                 SetServiceDomainRequest
              </requestname>
              <userJID>tx1@localhost/tx1</userJID>
              <requestNID>tx1@localhost/tx1/3373203</requestNID>
              <state>8</state>
              <clientdata>RequestToken2</clientdata>
            </header>
            <data>
        <!-- Response data goes here -->
          <SetServiceDomainResponseData>
                 <!-- SetServiceDomainResponseData payload goes here
-->
         </SetServiceDomainRequestData>
            </data>
 </Response>
```

Generic Agent Message Data For Event Type Agent Messages

An event can be an info event or an error event. This is indicated by the state element in the payload

Example 2.4. Event

```
  <AgentEvent>
      <header>
            <eventLevel>
          <!-- Event level goes here:  -->
                200
            </eventLevel>
            <eventAgentJID>srcrelay1@localhost/srcrelay1</
eventAgentJID>
                <eventWallclock>1288358866404</eventWallclock>
            <eventname>
           <!-- Event Name goes here: SourceEvent/StreamStatusEvent --
>
              SourceEvent
            </eventname>
        </header>
   <data>
     <!-- Event data goes here -->
     <SourceEventData>
          <!-- Source Event data payload goes here -->
     </SourceEventData>
   </data>
   <log/>
  </AgentEvent>
```

Chapter 3. Generic Agent API Overview

The following set of operations will be typically performed by the devices hosting the agents outside of the agent API

Startup Operations

XMPP Login

First the Client does a standard XMPP Login. The credentials needed for this need to be provided to the Client by the Administrator by some out-of-band means.

- Username@DomainName.com (Who)
- Password
- XMPPResourceID (From Where)

SetServiceDomain

After the base XMPP login the device has to discover the contactJID for the AppServerAgent The AppServerAgent device publishes its userJID as the contactJID for the AppServerAgent based on static configuration on the device hosting the AppServerAgent. The Device then does a "SetServiceDomain" which registers the Device with the AppServerAgent.

- The registration process includes publishing its userJID as the contactJID for the XMPPResource
- Device version number and licensing is also checked at this time
- The Device version number can be obtained from the Version section of the API document corresponding to the client implementation

This will change the status to not ready indicating that it has logged in but cannot participate in a session as a source or destination.

Example 3.1. SetServiceDomainRequest extended from AgentRequest

```
<SetServiceDomainRequestData>
          <domain>default</domain>
          <AssetTemplateTitle>dcpc1</AssetTemplateTitle>
          <AssetPrivateData></AssetPrivateData>
          <AssetPrivateKeyData></AssetPrivateKeyData>
          <ServiceVersion>2.25</ServiceVersion>
          <IPAddress>10.1.2.190</IPAddress>
</SetServiceDomainRequestData>
```

If all is well, the response will contain the Device's private data(preferences) that has been configured/ stored on the server.

- The current date/time is also returned

Example 3.2. Service Login Response extended from AgentResponse

```
  <SetServiceDomainResponseData>
    <AssetResource NID="default.assetresourcelist.arl_a3f6cdf7-
be03-416d-8e0f-1a964edb60b0" Persistence="Temporary" Rev="1" Ver="1.7"
 parentNID="default.assetresourcelist" serviceVer="2.25">^M
    <Contact>
        <OwnerUserJID/>
        <Presence>unavailable</Presence>
    </Contact>
    <State>
        <StateFlag>Offline</StateFlag>
        <StateMsg/>
    </State>
    <TemplateAssetResourceNID/>
    <Info>
        <Groups>
            <Tag Name="Department" Value="Engineering"/>
        </Groups>
        <Title>dcpc1</Title>
        <Description/>
        <Type>IPVSClient-Device</Type>
        <AssetPrivateKey>
            <KeyScheme/>
            <KeyData/>
        </AssetPrivateKey>
        <AssetPrivateData/>
        <FeatureList>
            <Feature Name=""/>
        </FeatureList>
    </Info>
    </AssetResource>
    <PortResourceNIDList>
      <resourceNID
 type="MediaStreamDstPortResource">default.mediastreamdstportresourcelist.msdprl_4
ac35-4c46d3dc0177</resourceNID>
    </PortResourceNIDList>
    <deviceNID>default.devicelist.dl_236f39ca-2935-4365-
a624-95637cbcf4f9</deviceNID>
    <appServerAgentJID>srcrelay1@localhost/srcrelay1</
appServerAgentJID>
  </SetServiceDomainResponseData>
```

GetMediaRoomInvites

Once this response is received the Agent will send out a "GetMediaRoomInvitesRequest" (refer Session API document)

- This will publish the userJID as the contactJID for the MediaSource and MediaDestination resources associated with this device

- The AppServerAgent will use this contactJID when sending messages to agents associated with the resource
- This will also result in the AppServerAgent sending out invites to the device for all the MediaRooms that include the resources for the device

Chapter 4. MediaRoom Agent API

This chapter provides details on the different Agents and the set of messages that will be exchanged between agents as part of the MediaStream setup/control/teardown functions

AppServerAgent

This is the agent responsible for all MediaRoom setup and control functions. This has the following main functional areas:

- A Database of all active MediaRooms and their associated objects:

  MediaTrack: StreamTrack, RecordTrack
  MediaObjects: MediaSources, MediaDest, MediaRelays
  Streams: The streams are setup between MediaSources and MediaDestinations and my involve one or more MediaRelays

- A centralized run-time MediaRoom Co-ordinator that manages:

  MediaRoom Setup Requests
  Stream Setup Requests: A request is sent to DestStreamAgent associated with the MediaDest
  MediaObjects: Manage invitations and presense for Media Objects
  MediaTimelines Updates: A request is sent to SourceStreamAgent associated with the MediaSource

DestStreamAgent

This is the agent responsible for all MediaStream setup/teardown functions. This has the following main functional areas:

- Handling requests to setup/teardown streams

  The request includes all the information required to be setup the MediaSource and MediaDestinations to setup/teardown the stream
  Once the watch is received by the DestStreamAgent, it is now responsible for the Stream
  The Watch includes Setup. The UnWatch includes Teardown
  If the Setup fails, the corresponding state(SETUP_ERROR) is maintained in the DestStreamAgent
  A Setup/Teardown of the stream results in Start/Stop Stream requests being sent to the SourceStreamAgent

Example 4.1. Generic DestStreamAgent Request payload received from the AppServerAgent extended from AgentRequest

```
<ActionRequestData>
  <action>
<!-- action goes here -->
      Watch
  </action>
  <streamNID>3bebcbb1-3ead-47b7-b006-a1c7f829b34e</streamNID>
  <streamdatadoc>
<!-- Stream object goes here -->
  </streamdatadoc>
  <mediasourcedatadoc>
<!-- MediaSource object goes here -->
  </mediasourcedatadoc>
</ActionRequestData>
```

Example 4.2. Stream Setup Request payload received from the AppServerAgent extended from Generic DestStreamAgent Request

```
<ActionRequestData>
  <action>Watch</action>
  ...
</ActionRequestData>
```

Example 4.3. Stream Teardown Request payload received from the AppServerAgent extended from Generic DestStreamAgent Request

```
<ActionRequestData>
  <action>UnWatch</action>
  ...
</ActionRequestData>
```

- Maintaining a state machine for each of the MediaStreams which is updated based on

  The presence of the MediaSource
  The Start/Stop Response AgentMessages received from the SourceStreamAgent
  The Update Response AgentMessages received from the SourceStreamAgent
  The SourceEvent AgentEvents received from the SourceStreamAgent

- The state machine action handlers are responsible for

Restarting a stream once an absent MediaSources comes back into the MediaRoom
Sending out StreamStatusUpdates whenever the state of the stream changes

Example 4.4. StreamStatusUpdate payload sent to the AppServerAgent extended from AgentEvent

```
  <StreamStatusEventData roomNID="136cc62f-6537-4cfd-
b677-8d09d37738e2" streamNID="3bebcbb1-3ead-47b7-b006-a1c7f829b34e">
    <state>
       <!-- Stream state value goes here -->
       6
    </state>
    <lastEventData>
  <!-- Last Event Data goes here -->
    </lastEventData>
    <relayData relayRoomNID="" relayStreamNID=""/>
  </StreamStatusEventData>
```

SourceStreamAgent

This is the agent responsible for all MediaStream setup/teardown/control functions. This has the following main functional areas:

- Handling start/stop request for streams from the DestStreamAgent

  The DestStreamAgent sends a Start request with all the source and dest information
  The dest information is used to setup output pins in the streaming server
  The source information is used to setup input pins in the streaming server
  Multiple dest-output pins can be connected to a single source-input pin in the streaming server
  The source-input pin are released once there are no more dest-output pins connected to it

Example 4.5. SourceStreamAgent Action Request payload received by the SourceStreamAgent extended from AgentRequest

```
  <ActionRequestData>
<!-- MediaRoomServiceUtils.createActionRequestMessage -->

        <streamNID>3bebcbb1-3ead-47b7-b006-a1c7f829b34e</streamNID>
        <useraction>
         <!-- User action goes here -->
           Start
        </useraction>
        <action>
         <!-- Action goes here -->
        StartRequest
        </action>
        <actiondata>
         <!-- ActionData goes here -->
        </actiondata>
        <wallClock>
         <!-- WallClock at which this action was fired -->
           0
        </wallClock>
        <actionNID>
         <!-- A unique identified for the action -->
           136cc62f-6537-4cfd-b677-8d09d37738e2
        </actionNID>
        <streamdatadoc>
         <!-- Stream object goes here -->
        </streamdatadoc>
        <mediasourcedatadoc>
         <!-- One or more MediaSource object(s) go here -->
        </mediasourcedatadoc>
  </ActionRequestData>
```

Example 4.6. Stream Start Action Request action data received by the SourceStreamAgent extended from Generic SourceStreamAgent Request

```
  <ActionRequestData>
              ...
        <useraction>Start</useraction>
        <action>StartRequest</action>
        ...
  </ActionRequestData>
```

Example 4.7. Stream Stop Action Request received by the SourceStreamAgent extended from Generic SourceStreamAgent Request

```
  <ActionRequestData>
        <useraction>Stop</useraction>
        <action>StopRequest</action>
        ...
  </ActionRequestData>
```

Example 4.8. Generic SourceStreamAgent Response payload sent by the SourceStreamAgent

```
  <ActionResponseData>
        <streamNID>3bebcbb1-3ead-47b7-b006-a1c7f829b34e</streamNID>
        <useraction>
       <!-- User action goes here -->
          Start
        </useraction>
        <action>
       <!-- Action goes here -->
          StartRequest
        </action>
        <actiondata>
       <!-- ActionData goes here -->
        </actiondata>
        <actionNID>
       <!-- A unique identified for the action -->
          136cc62f-6537-4cfd-b677-8d09d37738e2
        </actionNID>
        <streamURL>v2d://10.1.103.1/port=6060?bandwidth=10240000?
avoption=Video?ismulticast=false?enablekbm=false</streamURL>
        <relayData relayRoomNID="" relayStreamNID=""/>
        <startTime>1288453969469</startTime>
        <status>200</status>
        <log/>
  </ActionResponseData>
```

Example 4.9. Stream Start Response sent by SourceStreamAgent to the DestStreamAgent extended from Generic SourceStreamAgent Response

```
 <ActionResponseData>
       <useraction>
       Start
       </useraction>
       <action roomNID="" sync="false" syncAgentJID=""
syncSourceParentDirNID="" syncSourceUUID="" syncTC="">
       StartRequest
       </action>
       ...
 </ActionResponseData>
```

Example 4.10. Stream Stop Response sent by SourceStreamAgent to the DestStreamAgent extended from Generic SourceStreamAgent Response

```
 <ActionResponseData>
       <useraction>
       Stop
       </useraction>
       <action roomNID="" sync="false" syncAgentJID=""
syncSourceParentDirNID="" syncSourceUUID="" syncTC="">
       StopRequest
       </action>
       ...
 </ActionResponseData>
```

- Handling update timeline requests for streams from the AppServerAgent

  The AppServerAgent sends an TimelineUpdate request with all the source information for the new offset
  The new source information is used to setup new input pins in the streaming server
  The existing dest-output pins are switched to the new source-input pins in the streaming server
  The source-input pin are released once there are no more dest-output pins connected to it

Example 4.11. Stream TimelineUpdate Request AppServerAgent to the SourceStreamAgent

```
<TimelineUpdateRequestData>
<data>
 <action/>
<!--  Pause(StopOutput), Resume(SwitchLookback/UpdateOffset),
 Catchup(StartOutput/SwitchDefault)-->
 <actiondata>
  <timelineupdate>
   <MediaTimeline/>
  </timelineupdate>
 </actiondata>
 <actionNID/>
</data>
</TimelineUpdateRequestData>
```

Example 4.12. Stream TimelineUpdate Response sent by SourceStreamAgent to the DestStreamAgent extended from Generic SourceStreamAgent Response

```
  <ActionResponseData>
        <useraction>
        SkipForward
        </useraction>
        <action roomNID="" sync="false" syncAgentJID=""
 syncSourceParentDirNID="" syncSourceUUID="" syncTC="">
        UpdateOffset
        </action>
        <actiondata>
        120
        </actiondata>
        ...
  </ActionResponseData>
```

- It is responsible for generating the following SourceEvents when notifications are received from the streaming server

  Info Event: Generated when the first bits of data are available on the stream
  Error Event: Generated when the there is any error in the stream

Example 4.13. Source Event sent by SourceStreamAgent to the DestStreamAgent extended from Generic AgentEvent

```
  <SourceEventData>
        <eventCode>
        <!-- Event Specific Code goes here -->
          110
        </eventCode>
        <action>SourceEvent</action>
     <streamNID>02301fd9-ed29-40eb-b1b6-84ce358bf30a</streamNID>
     <startTC>0</startTC>
     <sourceEventActionData>
       <!-- Event Specific Data goes here -->
       srcrelay1@localhost/srcrelay1/Start/1288358866396
     </sourceEventActionData>
     <streamNIDList>
      <streamNIDListItem>3c85affb-ad7a-4213-9b6f-2e1ec1a134e1</
streamNIDListItem><streamNIDListItem>02301fd9-ed29-40eb-
b1b6-84ce358bf30a</streamNIDListItem>
     </streamNIDList>
     <status>200</status>
  </SourceEventData>
```

- Setup/Teardown of any upstream MediaStreams if the actual source is not local but is using an relay stream to an upstream SourceStreamAgent

  The relay stream is managed by the DestStreamAgent co-located on the device
  The co-located DestStreamAgent propogates all relay stream state changes which are Stream state changes which result in SourceEvents

Appendix A. Data Model

The following are sample XMLs that reflect the structure of Objects involved in the MediaRoom service.

MediaRoom Objects

The MediaRoom Objects are listed below. The corresponding xml is available under the svn. The base svn URL is "http://192.168.1.12/svn/svnroot/model/trunk/mediaroomXML/com/ipvs/mediaroomservice/datamodel/" To get the specific xml file append the Filename mentioned next to the Object

MediaRoom : MediaRoomInstance.xml
MediaTrack : MediaTrackInstance.xml
MediaSource : MediaSourceInstance.xml
MediaDest : MediaDestInstance.xml
Stream : StreamInstance.xml

MediaRoom API Messages

The MediaRoom Messages are listed below. The corresponding xml is available under the svn. The base svn URL is "http://192.168.1.12/svn/svnroot/model/trunk/mediaroomXML/com/ipvs/mediaroomservice/agent/" To get the specific xml file append the Filename mentioned next to the Object

DestStreamAgentRequest : DestStreamAgentRequest.xml
SourceStreamAgentRequest : SourceStreamAgentRequest
SourceStreamAgentResponse : SourceStreamAgentResponse
SourceEvent : SourceEventInstance.xml
StreamStatusEvent : Uses the Status element in the StreamInstance object