RTMP stream capturing by re-publishing from another RTMP server

Overview

WCS can capture RTMP stream translated by another RTMP server. Technically, the capture process of republished stream does not differ from stream capture using RTMP encoder or Flash application. Let's consider below Adobe Media Server as RTMP stream source for WCS.

AMS setup to stream republishing

Adobe Media Serveris a server software for live streaming targeted the clients using Adobe Flash Player. By default, server allows to publish a stream, so a special application should be made for re-publishing.

1. Consider AMS installation on Linux server in /opt/adobe/ams directory. Server applications are placed to applications subdirectory. Make the republish application directory:

```
cd /opt/adobe/ams/applications
mkdir republish
```

2. Make the application script file main.asc in /opt/adobe/ams/applications/republish directory

Script variables setup:

```
var wcsServer = "192.168.0.5";
var netConnections = new Object();
var streams = new Object();
var roomName = "#amsroom1";
```

Here

- wcsServer is the WCS server address to republish:
- roomName is the suffix to add to the stream name for WCS server.

Publisher connection handling. Here connection to WCS server is established for pepublishing:

```
application.onConnect = function (client){
    trace("onConnect "+client.id);
    var nc = new NetConnection();
    nc.ping = function(){
        nc.call("pong",null);
    }
    nc.connect("rtmp://"+wcsServer+":1935/live");
    nc.onStatus = function(info){
        trace("onStatus info.code: "+info.code);
        if (info.code=="NetConnection.Connect.Success"){
            trace("connection opened: "+wcsServer);
        }
    }
    netConnections[client.id]=nc;
    trace("onConnect done");
    return true;
}
```

Stream publishing handling. Here the stream is republishing to WCS server with suffix addition to the stream name:

```
application.onPublish = function(client, myStream){
    var wcsStreamName = myStream.name+roomName;
    trace("onPublish "+myStream.name+" by client.id "+client.id);
    var nc = netConnections[client.id];
    var ns = new NetStream(nc);
    ns.onStatus = function(info){
        if (info.code == "NetStream.Publish.Start"){
            trace("now publishing "+myStream.name);
        }
    }
    ns.attach(myStream);
    ns.publish(wcsStreamName);
    streams[myStream.name]=ns;
    trace("published stream "+wcsStreamName+" to: "+wcsServer);
    ns.publish(false);
    ns.publish(wcsStreamName);
}
```

Stream publish stopping handling. Here republishing the stream to WCS server stops:

```
application.onUnpublish = function(client, myStream){
    trace("onUnpublish "+myStream.name+" by client.id "+client.id);
    var ns = streams[myStream.name];
    if (ns){
        ns.publish(false);
        var s = Stream.get(myStream.name);
        Stream.destroy(s);
        delete streams[myStream.name];
        trace("unpublished "+myStream.name);
    }
}
```

Publishers' connection closing handling. Here WCS server connection is closing:

```
application.onDisconnect = function (client){
    trace("onDisconnect "+client.id);
    var nc = netConnections[client.id];
    if (nc){
        nc.close();
        delete netConnections[client.id];
        trace("disconnected "+client.id);
    }
}
```

AMS republish application script

```
var wcsServer = "192.168.0.5";
var netConnections = new Object();
var streams = new Object();
var roomName = "#amsroom1";

application.onConnect = function (client){
    trace("onConnect "+client.id);
    var nc = new NetConnection();
    nc.ping = function(){
        nc.call("pong",null);
    }
    nc.connect("rtmp://"+wcsServer+":1935/live");
    nc.onStatus = function(info){
        trace("onStatus info.code: "+info.code);
        if (info.code=="NetConnection.Connect.Success"){
            trace("connection opened: "+wcsServer);
        }
    }
    netConnections[client.id]=nc;
    trace("onConnect done");
    return true;
}

application.onDisconnect = function (client){
    trace("onDisconnect "+client.id);
    var nc = netConnections[client.id];
    if (nc){
        nc.close();
        delete netConnections[client.id];
        trace("disconnected "+client.id);
    }
}

application.onPublish = function(client, myStream){
    var wcsStreamName = myStream.name+roomName;
    trace("onPublish "+myStream.name+" by client.id "+client.id);
    var nc = netConnections[client.id];
    var ns = new NetStream(nc);
    ns.onStatus = function(info){
        if (info.code == "NetStream.Publish.Start"){
            trace("now publishing "+myStream.name);
        }
    }
    ns.attach(myStream);
    ns.publish(wcsStreamName);
    streams[myStream.name]=ns;
    trace("published stream "+wcsStreamName+" to: "+wcsServer);
    ns.publish(false);
    ns.publish(wcsStreamName);
}

application.onUnpublish = function(client, myStream){
    trace("onUnpublish "+myStream.name+" by client.id "+client.id);
    var ns = streams[myStream.name];
    if (ns){
        ns.publish(false);
        var s = Stream.get(myStream.name);
        Stream.destroy(s);
        delete streams[myStream.name];
        trace("unpublished "+myStream.name);
    }
}
```

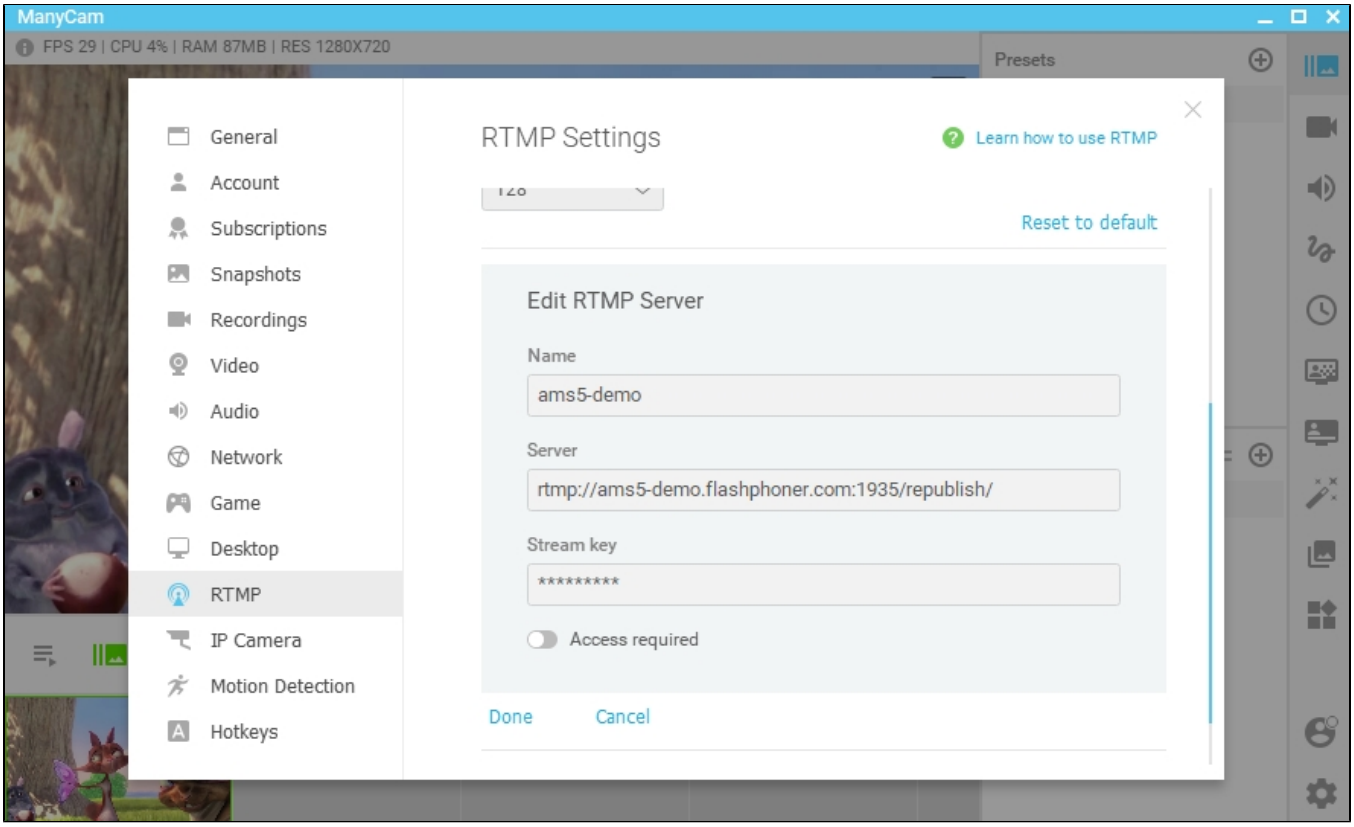
The application will be available on AMS server by URLrtmp://youramsserver:1935/republish, where youramsserver is your AMS server hostname.

Testing

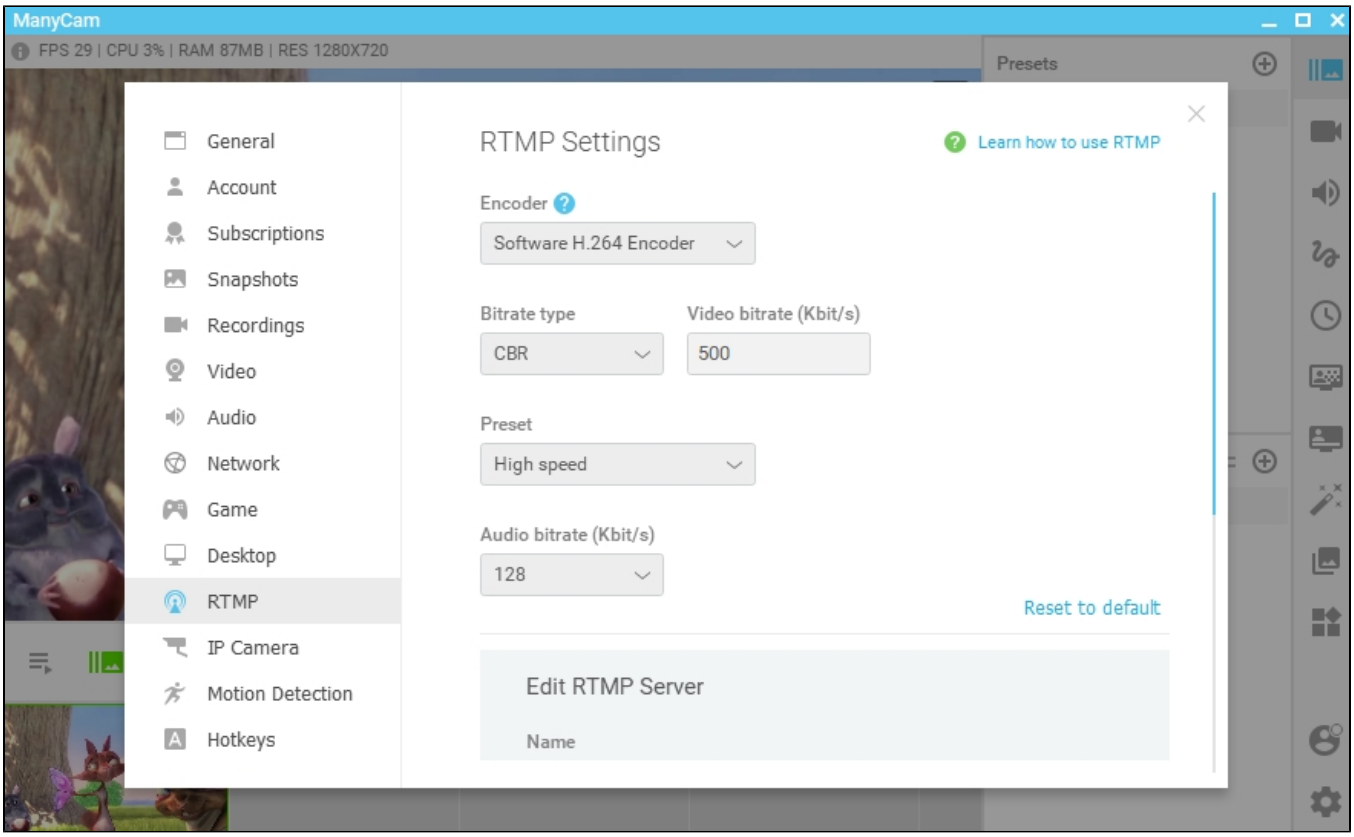
1. For test we use:

- AMS serverams5-demo.flashphoner.com
- WCS servermixer-demo.flashphoner.com
- ManyCam Virtual webcam to publish RTMP stream to AMS
- Playerweb application to playback the stream captured on WCS server

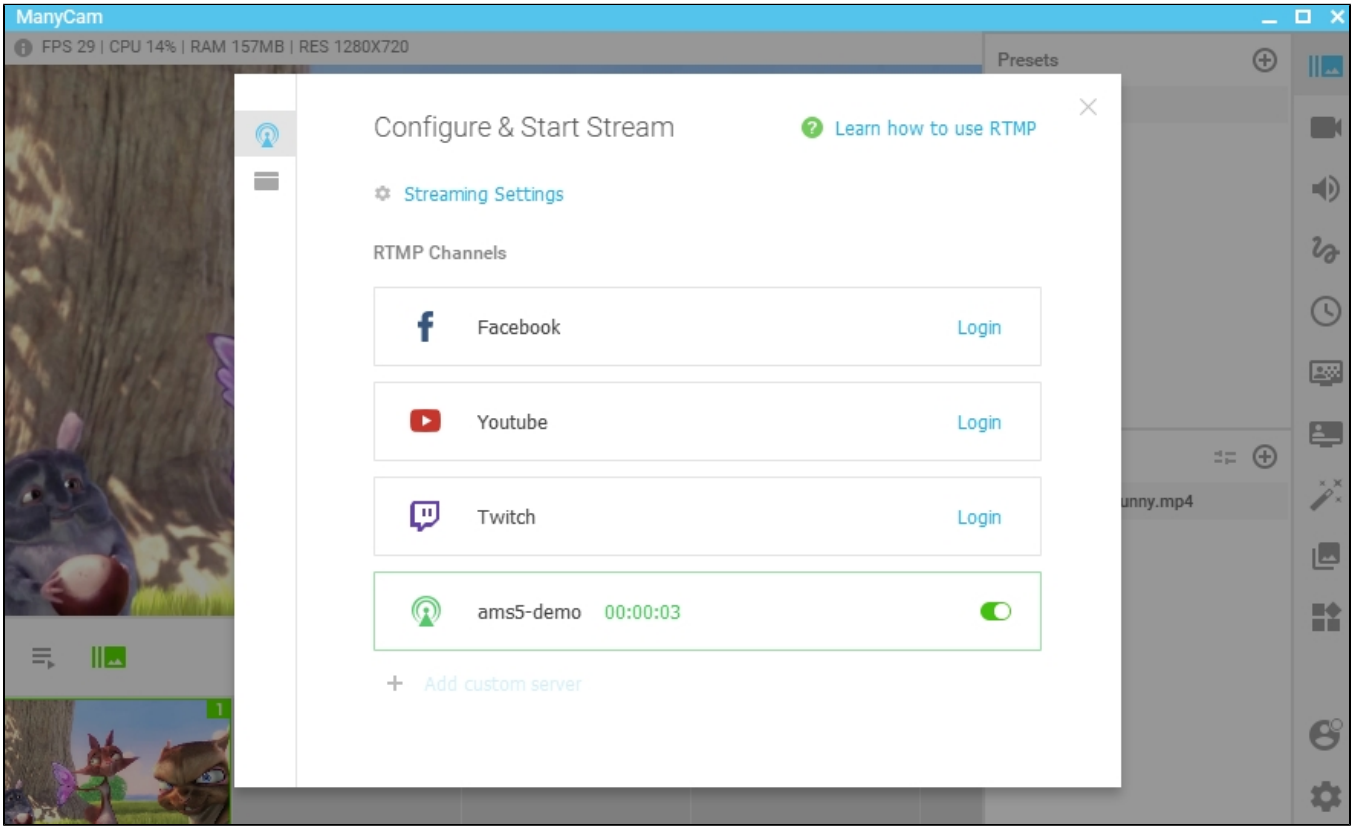
2. Set RTMP server parameters in ManyCam, set stream name to amsStream

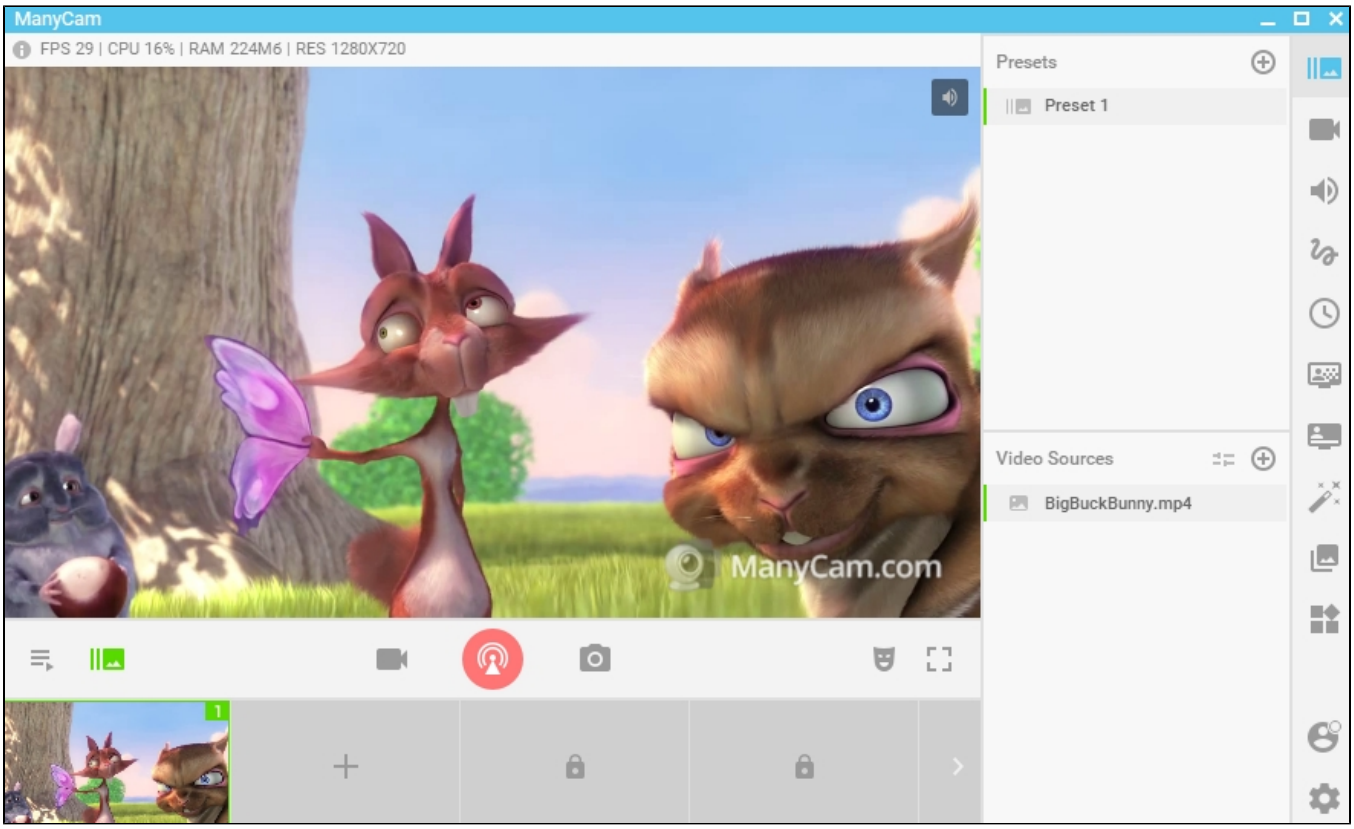


3. Set RTMP streaming parameters in ManyCam and press 'Done'



4. Start streaming from ManyCam





5. Open Player web application on WCS server. Set the stream name amsStream#amsroom1 in 'Stream' field and press 'Start'. The stream captured playback begins

Player

WCS URL	wss://mixer-demo.flashphoner.co
Stream	amsStream#amsroom1
Volume	