Flash Media Server Solutions List

Dennie Hoopingarner
Michigan State University
hooping4@msu.edu

Typeface conventions:

Section Headers

FMS objects, methods and properties

Actionscript code

Returned messages

Create application space on the server for a new application

Scenario: You want a separate space on the server for an FMS application.

Solution: Create a new folder on the server for the application.

Details: You must use the system file management to create a new folder under the server's "applications" folder. On a Linux machine, for instance, FMS is typically located in the /opt/macromedia/FMS directory. The applications are located under the FMS directory, visually represented thusly:

```
opt
  macromedia
    fms
      applications
```

Every application has its own folder under the "applications" folder. Everything above the "applications" folder level is invisible to FMS. Within the application's folder, a subfolder must be created called "streams." Under the "streams" folder, sub-folders can be created to distinguish instances of the application.

```
opt
  macromedia
    fms
      applications
        yourapp
          streams
            roomA
            roomB
            roomC
            Dansroom
        myapp
          streams
            room1
            room2
            testroom
```

When you attach to the server using a **NetConnection** object, you map to the specific application AND the application's instance, but SKIP referring to the "streams" folder level. For example, using the sample architecture above, if you wanted to map to room2 of myapp, you would use the following argument of the **connect()** command:

```
myNC.connect("rtmp://my.server.name/myapp/room2");
```

Note: on Linux systems, you must make sure that the owner of the application can read and write. When you install the server, the installer prompts you to create a default user, which will be something like "nobody" or "fms."

The syntax to change the ownership:

sudo chown -R *user application*

Connect to the server using the NetConnection Object

Scenario: you want to make a connection to the FMS.

Solution: use the **NetConnection** object's **connect** command.

Details: Type this Actionscript code in the frame's timeline:

```
myNC = new NetConnection();
myNC.onStatus = function(info){
        trace(info.code);
}
myNC.connect(rtmp://path.to.your.server/application);
```

Notes: The **onStatus** call is optional, but is recommended during development for testing purposes. If your code makes a connection successfully, the following message will be returned in the output window:

NetConnection.Connect.Success

Create a stream to play/record audio using the NetStream Object

Scenario: you want to use a **NetConnection** to stream audio up and/or down between client and server.

Solution: attach a **NetStream** object to the **NetConnection** object.

Details: Type this Actionscript code in the frame's timeline:

```
myNC = new NetConnection();
myNC.connect(rtmp://path.to.your.server/application/instance);
myNS = new NetStream(myNC);
myNS.onStatus = function(info){
        trace(info.code);
}
```

Notes: The **onStatus** call is optional, but it is recommended during development for testing purposes. When you use the NetStream to play or record audio, messages will be returned in the output window.

Play a MP3/FLV audio clip

Scenario: You want to stream an audio clip from the server to the client.

Solution: Use the **NetStream.play()** command.

Details: Create a "play" button called play_btn. Create a "stop" button called stop_btn. Type this script in the frame's timeline:

```
myNC = new NetConnection();
myNC.connect(rtmp://path.to.your.server/application/instance);
myNS = new NetStream(myNC);
myNS.onStatus = function(info){
        trace(info.code);
}

play_btn.onRelease = function(){
        // If you want to play a FLV file:
        myNS.play("yourFile");
        //If you want to play a MP3 file:
        myNS.play("mp3:yourFile");
}

stop_btn.onRelease = function(){
        myNS.play(false);
}
```

Attach the system microphone to capture and record audio

Scenario: You want to be able to capture audio input from your microphone.

Solution: Use the **Microphone.get()** command, attach the Microphone object to a **NetStream** object.

Details: Type this code into the frame's timeline:

```
myNC = new NetConnection();
myNC.connect(rtmp://path.to.your.server/application/instance);
myNS = new NetStream(myNC);
myNS.onStatus = function(info){
        trace(info.code);
}
myMic = Microphone.get();
myNS.attachAudio(myMic);
```

Record and play back audio

Scenario: You want to record and play back an audio clip.

Solution: Use the **NetStream** object's **publish()** and **play()** functions.

Details: Create buttons to record, stop, and play audio. Name them record_btn, stop_btn, and play_btn. Add this code to the frame's timeline:

```
myNC = new NetConnection();
myNC.connect(rtmp://path.to.your.server/application/instance);
myNS = new NetStream(myNC);
myNS.onStatus = function(info){
        trace(info.code);
}
myMic = Microphone.get();
myNS.attachAudio(myMic);

record_btn.onRelease = function(){
        myNS.publish("mySound", "record");
}
stop_btn.onRelease = function(){
        myNS.publish(false);
        myNS.play(false);
}
play_btn.onRelease = function(){
        myNS.play("mySound");
}
```

Attach the system camera and microphone to capture audio and video

Scenario: You want to be able to capture video from the system's attached camera.

Solution: Use the **Microphone.get()** and **Camera.get()** commands, attach the Microphone object to a **NetStream** object, and attach the camera's video output to a video object on the stage.

Details: Type this code into the frame's timeline:

```
myNC = new NetConnection();
myNC.connect(rtmp://path.to.your.server/application/instance);
myNS = new NetStream(myNC);
myNS.onStatus = function(info){
        trace(info.code);
}
myMic = Microphone.get();
myNS.attachAudio(myMic);
```

Record and playback audio/video

Scenario: You want to record and play back an audio/video clip.

Solution:

Details: Create buttons to record, stop, and play audio. Name them record_btn, stop_btn, and play_btn. Add this code to the frame's timeline:

```
myNC = new NetConnection();
myNC.connect(rtmp://path.to.your.server/application/instance);
myNS = new NetStream(myNC);
myNS.onStatus = function(info){
        trace(info.code);
}
myMic = Microphone.get();
myNS.attachAudio(myMic);

record_btn.onRelease = function(){
        myNS.publish("mySound", "record");
}
stop_btn.onRelease = function(){
        myNS.publish(false);
        myNS.play(false);
}
play_btn.onRelease = function(){
        myNS.play("mySound");
}
```

Create a persistent shared object to store text data using the SharedObject object

Scenario: You want to be able to save text data to the server.

Solution: Create a Shared Object to store strings of text.

Details: Calling a Shared Object will access an existing SO, or if it doesn't exist, the command will create the SO. For example, if you want to store data in a variable called "myList," use this code:

```
init();
stop();
function init(){
        myNC = new NetConnection();
        myNC.connect("rtmp://path.to.server/nameOfApplication");
        so = SharedObject.getRemote("myList", myNC.uri, true);
        so.onSync = function(){
                //call a function to handle your data
        }
        so.connect(myNC);
}
```

Save SharedObject data

Scenario: You want to save text data to the server.

Solution: Use a Shared Object and store the text in it.

Details: Create a Shared Object, then define a "slot" to store your text data.

For example, if your Flash application has a list of users in an array called studentArray, you can store them in a Shared Object called myStudents, in a slot called studentNames.

```
myNC = new NetConnection();
myNC.connect("rtmp://path.to.server/nameOfApplication");
so = SharedObject.getRemote("myStudents", myNC.uri, true);
so.connect(myNC);
```

Use this operation to create or update the student list:

```
so.data.studentNames = studentArray;
```

Retrieve SharedObject data

Scenario: You want to load the text data that you have saved to the server.

Solution: retrieve properties from the application's Shared Object.

Details: After you have made a connection to the server and made reference to the Shared Object, you can retrieve its data. If, for example, you have a list of users in an array called myList, calling the command:

```
userList = so.data.myList
```

creates an array in your Flash document called userList, containing your list of users.

Delete a stream

Scenario: You want to delete an audio or video clip from the server.

Solution: Re-record the file with a length of zero.

Details: This code demonstrates how to use the NetStream.info object to detect when the recording has started. As soon as recording starts, the code stops recording. FMS doesn't allow a zero-length file, so the server will delete the file from the server.

```
init();
stop();

function init()
	mySound = "littleOleSound";
	delMe = false;
	myNC = new NetConnection();
	myNC.onStatus = function(info){
		trace(info.code);
	}
	myNC.connect("rtmp://path.to.server/nameOfApplication");
	myNS = new NetStream(myNC);
	myNS.onStatus = function(info){
		trace(info.code);
		if((info.code=="NetStream.Publish.Start") && (delMe==true)){
			myNS.publish(false);
			delMe = false;
		}
	}
	myMic = Microphone.get();
	myMic.setRate(22);
	myNS.attachAudio(myMic);
}

record_btn.onRelease = function(){
	myNS.publish(mySound, "record");
}
stop_btn.onRelease = function(){
	myNS.publish(false);
	myNS.play(false);
}
play_btn.onRelease = function(){
	myNS.play(mySound);
}
delete_btn.onRelease = function(){
	delMe = true;
	myNS.publish(mySound, "record");
}
```

Retrieve a stream's length

Scenario: You want to return the length of a recorded clip.

Solution: Use the metadata of the stream to return the "duration" value.

Details: As soon as a stream begins playing, the code can return the stream's metadata. The code snippet below shows where to call the metadata.

```
        myNC = new NetConnection();
        myNC.onStatus = function(info){
            trace(info.code);
        }
        myNC.connect("rtmp://path.to.server/nameOfApplication");
        myNS = new NetStream(myNC);
        myNS.onMetaData = function(info){
            trace(info.duration);
        }
}
```

Label clips with a user's ID using a SharedObject

Scenario: You want to be able to identify the creator of a recorded clip.

Solution: Retrieve the user's identification from the Shared Object, then label the recording with the ID code.

Details: One way of storing student information is in an array. The array can be an item in an Shared Object. To access the student's info, retrieve the contents of a Shared Object, then parse the array to find the student's data.

```
myNC = new NetConnection();
myNC.onStatus = function(info){
        trace(info.code);
        if(info.code == "NetConnection.Connect.Success"){
                mySO = SharedObject.getRemote("studentNames", myNC.uri, true);
                mySO.onSync = function(){
                        for (x=0; x<mySO.data.studentNames.length; x++){
                                if(mySO.data.studentNames[x]=="myName"){
                                        trace("Logged in");
                                }
                        }
                }
                mySO.connect(myNC);
        }
}
myNC.connect("rtmp://path.to.server/nameOfApplication");
```

Detect the end of playback with the NetStream information object

Scenario: You want to detect when a clip has finished playing.

Solution: Detect the succession of two NetStream events: Play.Stop and Buffer.Empty

Details: The code snippet below shows how to use the onStatus event of the NetStream object to detect the NetStream events.

```
myNS.onStatus = function(info){
        if (info.code == "NetStream.Play.Stop"){
                this.seenStop = true;
        }
        else if (info.code == "NetStream.Play.Start"){
                this.seenStop = false;
                }
        else if (info.code == "NetStream.Buffer.Empty" && this.seenStop){
                //
                //call your function here
                //
                this.close();
                this.seenStop = false;
        }
}
```

Play from or skip to a specific place in a clip using the seek() function

Scenario: You want to begin playing a clip at a specific time.

Solution: use the "start" parameter of the Netstream.play() method.

Details: For example, if you want to begin playing a clip called "mySound123" from 30 seconds after the beginning, use the following code:

```
myNS.play("mySound123", 30);
```

You can also play for a specific length of time. For example, if you want to play a clip from 15 seconds in to 45 seconds in (ora total playing time of 30 seconds), use this code:

```
myNS.play("mySound123", 15, 30);
```

Append to a recording

Scenario: You want to append to an existing audio/video clip.

Solution: use the "append" parameter of the Netstream.publish() method.

Details: Use the same code as for recording. Instead of the "record" parameter, use "append."

```
record_btn.onRelease = function(){
        myNS.publish(mySound, "append");
}
```