

## Creating a Custom FLV Player

So, you're tired of the same old video player? Solution, create your own. Included in this tutorial is all the script you need and source files to see things working.

### **Step 1**

Establish the connection with our .flv on a server. We do this instead of importing a movie and playing it in the timeline (never import a full video in your timeline).

Here's the code:

```
// Establishes a connection between Flash Player and a Flash Media Server
var nc:NetConnection = new NetConnection();
nc.connect(null);

// Establishes a connection between the Flash Player and the local files
var ns:NetStream = new NetStream(nc);
```

### **Step 2**

Set up the boundaries in which your streaming video will play within.

Here's the code:

```
// Sets up an empty video instance with the (width, height) defined
var vid:Video = new Video(320, 240);

// Adds the empty video instance to the stage
this.addChild(vid);
```

### **Step 3**

Attach the stream to the empty video instance. This takes the video box we've established when we added the child "vid" and tells it to hold the streaming video from the netConnection we've established in **Step 1**.

Here's the code:

```
// Sets the video stream to play within the empty video instance
vid.attachNetStream(ns);

// The location of the file hosted on your server. If using an external server it
may read: http://www.domain.com/videos/your_file.flv
ns.play("your_file.flv");
```

#### Step 4

Next we'll use Flash's built in classes to grab information and events on the video.

Here's the code:

```
// Sets a variable to hold information on the object defined.
var netClient:Object = new Object();

// onMetaData: used to receive descriptive information on the video loaded.
netClient.onMetaData = function(meta:Object)
{
    // returns the duration of the video
    trace(meta.duration);
};

// Establishes a connection between ns (the netStream) and netClient
(receiving descriptive information on the video) This allows the user to use
that information throughout coding the video player by referencing ns.
ns.client = netClient;

// Used to receive event messages
ns.addEventListener(NetStatusEvent.NET_STATUS, netstat);

function netstat(stats:NetStatusEvent) {
    trace(stats.info.code);
}
```

#### Step 5

Now to define some controls for our playback. First we'll set up a button for playing and pausing the video. If you're using the provided file those are on the stage and called mc\_play and mc\_pause.

Here's the code:

```
// These are basic MouseEvent functions. If you are unfamiliar with those
please refer to Week2: Making Buttons with ActionScript 3.0 at
school.smple.com
mc_play.addEventListener(MouseEvent.CLICK, playNS);
mc_pause.addEventListener(MouseEvent.CLICK, pauseNS);

function playNS(event:MouseEvent):void {
    // References the netStream and tells it to 'resume' playing from the
current location if it is paused.
    ns.resume();
}
```

```
function pauseNS(event:MouseEvent):void {
    // References the netStream and tells it to 'pause' at the current
    location
    ns.pause();
}
```

## Step 6

Let's make this Rewind or Fastforward now. I am using two movieclips called mc\_rewind and mc\_ff.

Here's the code:

```
// Tells the rewind button to start rewinding when the mouse is held down
and stop when it is released
mc_rewind.addEventListener(MouseEvent.CLICK, rewindNS);
mc_rewind.addEventListener(MouseEvent.CLICK, stopRewindNS);

// When the mouse is held down on the movieclip, this function will run
function keepRewinding(event:Event):void {
    // First we want to pause the video so it doesn't keep trying to play as
    we're rewinding it.
    ns.pause();
    /* Next we use netStreams 'seek' class which goes to any point in the
    movie specified in seconds. So, ns.seek(30); will go 30 seconds into a video.
    For the seek time on the rewind button we call ns.time to grab the
    current time of the video, then subtract 2(seconds) from it. As this is called
    continually, the time updates and keeps getting closer to 0.
    */
    ns.seek(ns.time - 2);
}

// When the mouse is pressed down on the movieclip, this function will run
function rewindNS(event:MouseEvent):void {
    // Adds an event listener to continually run while the mouse is held
    down
    mc_rewind.addEventListener(Event.ENTER_FRAME, keepRewinding);
    // Adds an event listener to check when the mouse goes outside of the
    movieclip
    mc_rewind.addEventListener(MouseEvent.CLICK,
    stopRewindNS);
}
```

```
// When the mouse is released from the movieclip, this function will run
function stopRewindNS(event:MouseEvent):void {
    // Both of the following lines remove the event listener from that
    // causes the video to rewind
    mc_rewind.removeEventListener(Event.ENTER_FRAME,
    keepRewinding);
    mc_rewind.removeEventListener(MouseEvent.MOUSE_OUT,
    stopRewindNS);
    ns.resume();
}

// All of this is the same as rewind, but in reverse. Instead of subtracting 2,
// we add 2 to the netStream seek.
mc_ff.addEventListener(MouseEvent.MOUSE_DOWN, ffNS);
mc_ff.addEventListener(MouseEvent.MOUSE_UP, stopFFNS);

function keepFastForwarding(event:Event):void {
    ns.pause();
    ns.seek(ns.time + 2);
}

function ffNS(event:MouseEvent):void {
    mc_ff.addEventListener(Event.ENTER_FRAME, keepFastForwarding);
    mc_ff.addEventListener(MouseEvent.MOUSE_OUT, stopFFNS);
}

function stopFFNS(event:MouseEvent):void {
    mc_ff.removeEventListener(Event.ENTER_FRAME,
    keepFastForwarding);
    mc_ff.removeEventListener(MouseEvent.MOUSE_OUT, stopFFNS);
    ns.resume();
}
```

## Step 7

Now we'll set a connection to the audio of the video playing and allow the user to mute/unmute it. We're using `mc_audio` for this with 2 frames set up inside of the movieclip. The first frame has a `stop()` action on it and a graphic of a speaker without audio. The second simply has a graphic of a speaker with audio.

Here's the code:

```
mc_audio.addEventListener(MouseEvent.CLICK, audioToggle);

// Sets a variable for the initial volume of the video.
var newVolume = .5;

// Creates a function to adjust the volume of the video.
function audioAdjust(event:Event):void {
    // Enables audio control and functionality
    var st:SoundTransform = new SoundTransform();
    // Ties the variable st to the volume of the video
    st.volume = newVolume;
    // Changes the volume of the video to the value of st (between 0
and 1, 0 being muted, 1 being full volume.
    ns.soundTransform = st;
}

// Calls the audioAdjust function with the initial volume at .5 (controlled by
the newVolume variable)
audioAdjust(null);

// Toggles the volume on/off.
function audioToggle(event:MouseEvent):void {
    // Checks to see the current frame of mc_audio. If it is on 1, then the
volume is set to 0 and the function audioAdjust is run to change the volume.
Then mc_audio is sent to frame 2.
    if (mc_audio.currentFrame == 1) {
        newVolume = 0;
        audioAdjust(null);
        mc_audio.gotoAndStop(2);
    } else {
        // If mc_audio is not on frame 1, then the volume is set to .5 and the
function audioAdjust is run to change the volume. Then mc_audio is sent to
frame 1
        newVolume = .5;
        audioAdjust(null);
        mc_audio.gotoAndStop(1);
    }
}
```

## Step 8

To extend this a bit, you can set up a small video player with multiple videos. Here is some sample script on how to switch between 2 videos. Granted if you're playing multiple videos I would recommend going the XML route, but for a down and dirty player, this will work perfect.

Here's the code:

```
test1.addEventListener(MouseEvent.CLICK, vid1);
test2.addEventListener(MouseEvent.CLICK, vid2);

function vid1(event:MouseEvent):void {
    // Removes the current video shell.
    this.removeChild(vid);
    // Adds a new video shell.
    this.addChild(vid);
    // Plays the file path given.
    ns.play('test1.flv');
}

function vid2(event:MouseEvent):void {
    this.removeChild(vid);
    this.addChild(vid);
    ns.play('test2.flv');
}
```