

Custom Video – ActionScript 3

Phase 1

When completed, your external video will play

```
//cable (likeTV) - allows for multiple streams like different channels
var ncConnect:NetConnection = new NetConnection();

//says is/is not a FlashComm video – rtmp if stream / null if not
ncConnect.connect(null);

//like a single channel - single stream to use
var nsStream:NetStream = new NetStream(ncConnect);

//new to AS3 - must declare client to listen to netstream!
nsStream.client = this;

//take video through stream and put in tv onstage
//new replacement to attachVideo() in AS2
vHolder.attachNetStream(nsStream);

//run the video
nsStream.play("cujo.flv");
```

Phase 2

When completed, Users can toggle play/pause and rewind video with buttons

Make controls layer and create base art for control strip. Convert this to a MovieClip with an Instance name of mcControls. In mcControls, make play, pause, and rewind buttons. Name them and place on their own layers in this control MovieClip – all at frame one.

Add the following code below existing code:

```
//run at once commands
var isPlaying:Boolean = true;
mcControls.btnPlay.visible = false;

//listeners
mcControls.btnPlay.addEventListener(MouseEvent.CLICK,pauseVideo);
mcControls.btnPause.addEventListener(MouseEvent.CLICK,pauseVideo);
mcControls.btnRewind.addEventListener(MouseEvent.CLICK,rewindVideo);

function pauseVideo(evtObj:MouseEvent):void{

    //note replacement to pause method below
    nsStream.togglePause();
    if(isPlaying == true){
        mcControls.btnPause.visible = false;
        mcControls.btnPlay.visible = true;
        isPlaying = false;
        //trace(isPlaying);
    }else{
        mcControls.btnPause.visible = true;
        mcControls.btnPlay.visible = false;
```

Custom Video – ActionScript 3

```
        isPlaying = true;
        //trace(isPlaying);
    }
}

function rewindVideo(evtObj:MouseEvent):void{

        nsStream.seek(0);

}
```

Phase 3

When completed, your external video now also has a download progress bar

Create *mcLoader* layer and draw loader rectangle/convert to mc and name **mcLoader_**. Put stroke or a fill on it's own layer and lock. This is the "always visible" duration indicator. On a different layer, create another fill (or use fill from inside stroke mentioned above) and convert it to a mc called *mcLoadbar* with instance name of **mcLoadbar** – remark at middle left!

Add the following code below existing code:

```
var vidTimer:Timer = new Timer(100,0); // 10 times a sec
vidTimer.addEventListener(TimerEvent.TIMER, videoStatus);
vidTimer.start();

var nPercentLoaded:Number;
var nBarLength:Number = mcControls.mcLoader.mcLoadbar.width;
//trace(nBarLength);

function videoStatus(evtObj:TimerEvent):void{

    nPercentLoaded = nsStream.bytesLoaded / nsStream.bytesTotal;
    mcControls.mcLoader.mcLoadbar.width = nPercentLoaded * nBarLength;

}
```

Phase 4

When completed, you create a scrubber handle that tracks with the video. At this point – the user cannot drag the handle. The handle will only move relative to the position in the video.

In the *mcLoader* movieclip, create a new *mcScrub* layer and draw the graphic you would like to be the handle. Convert this to a symbol and set *regmark* to center left. This also serves as the position indicator inside the loadbar. Name the instance *mcScrub* and set to the zero position (leftmost edge of loaderbar) Return to main timeline and open actions panel.

Modify the previous code as shown on next page:

Custom Video – ActionScript 3

At the top – just below new netStream constructor add the following and note you must reassign the client to metadata object

```
var nDuration:Number;
//-----metadata-----

// Build an object to handle metadata events
var metaHandler:Object = new Object();
metaHandler.onMetaData = function(meta:Object):void {

    nDuration = meta.duration;
    trace(nDuration);

}

// Assign the handler to the stream
nsStream.client = metaHandler;

//Note we're disabling this here! - Commented out
//nsStream.client = this;

function videoStatus():void{

    nPercentLoaded = nsStream.bytesLoaded / nsStream.bytesTotal;
    mcLoader.mcLoadbar._width = nPercentLoaded * nBarLength;
    mcControls.mcLoader.mcScrub.x = nsStream.time / nDuration * nBarLength;

}
```

Phase 5

When completed, Users can now drag the scrubber handle – remember that seek accuracy is dictated by the number of keyframes.

Note – all code below is new, written below video status. Begin by adding listeners for user drag actions. Next add rectangle constructor for boundary object to constrain drag. Write drag and stop drag functions. See examples below. Also note that the functions are toggling two timers – vidTimer and scrubTimer. One of these two timers is running but never at the same time.

```
mcControls.mcLoader.mcScrub.addEventListener(MouseEvent.CLICK, dragHandle);
mcControls.mcLoader.mcScrub.addEventListener(MouseEvent.CLICK, stopHandle);
```

```
var rect:Rectangle = new Rectangle(mcControls.mcLoader.mcLoadbar.x,mcControls.mcLoader.mcLoadbar.y,
mcControls.mcLoader.mcLoadbar.width, 0);
```

```
function dragHandle(evtObj:MouseEvent):void{

    vidTimer.stop();
    scrubTimer.start();
    mcControls.mcLoader.mcScrub.startDrag(false, rect);

    // down, now check for global mouseUp
    stage.addEventListener(MouseEvent.CLICK, stopHandle);

}
```

Custom Video – ActionScript 3

```
function stopHandle(evtObj:MouseEvent):void{

    vidTimer.start();
    scrubTimer.stop();
    mcControls.mcLoader.mcScrub.stopDrag();

    // be sure to clean up stage listener
    stage.removeEventListener(MouseEvent.MOUSE_UP, stopHandle);
}

var scrubTimer:Timer = new Timer(100,0); // 10 times a sec
scrubTimer.addEventListener(TimerEvent.TIMER, scrubStatus);

function scrubStatus(evtObj:TimerEvent):void{

    nsStream.seek(Math.floor((mcControls.mcLoader.mcScrub.x / nBarLength) * nDuration));
}
```