Der Quellcode zum Projekt

Livestream Connection Eventhandling – In der *VideoDisplay.as* wird der Live Stream connected:

```
private function connectLiveStream():void{
   _video = new Video();
   addChild( _video );
   _nc = new NetConnection();

   _nc.addEventListener(NetStatusEvent.NET_STATUS, netStatusConnection);
   _nc.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
                                                    securityErrorHandler);
   _nc.objectEncoding = ObjectEncoding.AMF0;

   _nc.client = new Object();
   _nc.client.streamisOver = onLiveStreamIsOver;
   _nc.connect(_url);
}

private function netStatusConnection(e:NetStatusEvent):void
  {
     switch(e.info.code)
     {
        case "NetConnection.Connect.Success":
           _ns = new NetStream( _nc );
           _ns.bufferTime = 3;
           dispatchEvent(new VideoEvent(VideoEvent.ON_PLAY, false, false ));
                    _ns.addEventListener(NetStatusEvent.NET_STATUS,
                                                              netLiveSatus);
                  _client = new Object();
           _client.onMetaData = onMetaData;
                    _ns.client = _client;
                    _video.attachNetStream( _ns );
           _ns.play(_streamId, -2, -1, 1);
                    break;
       case "NetConnection.Connect.Failed":
           dispatchEvent(new VideoEvent(VideoEvent.ON_LIVESTREAM_
                                                    FAILED, false, false));
           break;
     }
  }
```

Das Handling für eine fehlerhafte Connection wird im *MainController* abgehandelt:

```
.
.
...
_view.videoDisplay.addEventListener( VideoEvent.ON_LIVESTREAM_FAILED,
                                                    onLiveStreamFailed );

private function onLiveStreamFailed( e:VideoEvent ):void
  {
     // INFO: show popup with wmv-stream
          if( _model.isLivestreamActive == false ) return;
          _model.isLivestreamActive = false;
          _view.videoDisplay.removeEventListener( VideoEvent.ON_LIVESTREAM_
                                                    FAILED, onLiveStreamFailed );
          if( Capabilities.os.substr( 0, 3 ) != 'Mac' )
     {
        ExternalInterface.call( 'startVideo', _model.livestreamClipVO.wmvURL );
     }
  }
```

Die *VideoEvent*-Klasse abgeleitet von der *Event*-Klasse:

```
public class VideoEvent extends Event
{
    public static const NEW_TIME:String =
       'com.mbtv.events.VideoEvents.NEW_TIME';
     public static const ON_PLAY:String =
       'com.mbtv.events.VideoEvents.ON_PLAY';
     public static const ON_END:String =
       'com.mbtv.events.VideoEvents.ON_END';
   public static const ON_STOP:String =
       'com.mbtv.events.VideoEvents.ON_STOP';
   public static const ON_BUFFER_FULL:String =
       'com.mbtv.events.VideoEvents.ON_BUFFER_FULL';
   public static const ON_BUFFER_EMPTY:String =
       'com.mbtv.events.VideoEvents.ON_BUFFER_EMPTY';
   public static const ON_LIVESTREAM_FAILED:String =
       'com.mbtv.events.VideoEvents.ON_LIVESTREAM_FAILED';
   public static const ON_LIVESTREAM_END:String =
       'com.mbtv.events.VideoEvents.ON_LIVESTREAM_END';
   public static const ON_PREVIEW_LOADED:String =
       'com.mbtv.events.VideoEvents.ON_PREVIEW_LOADED';
       public var result:Object;
    public function VideoEvent(type:String, bubbles:Boolean=false, cancelable:
    Boolean=false, p_result:Object=null)
   {
       super(type, bubbles, cancelable);
       if( p_result != null) result = p_result;
   }
  }
}
```