Double Click Video Player set up.

for version 1.0.0

IMPORTS:

In the fla you only need to import two classes, the player and the video wrapper event class.
import com.blt.video.doubleClick.DCVideoPlayer;
import com.blt.events.BltVideoEvent;

LIBRARY IMPORTS:

From the component folder /DoubleClick Studio v2 AS3, include the following in the fla library only. Do not put them on the stage.

• **Video Player**
• **Video Streaming Library**

VIDEO PLAYER EVENTS:

All video events return a value for the current video (event.data.videoID) and current player (event.data.playerID). Video events only fire from the video player instance, they will not bubble. All your listeners must be attached directly to the player instance.

• **VIDEO_READY:** Fired when init is called.

• **VIDEO_PLAYING:** Fired every time video starts playing or resume playing.

• **PLAY_WITH_SOUND_CALLED:** Fired when replayWithSound is called.

• **VIDEO_MUTED:** Fired when muteAudio is called.

• **VIDEO_UNMUTED:** Fired when unmuteAudio is called.

• **VIDEO_PAUSED:** Fired when pauseVideo is called.

• **VIDEO_RESUMED:** Fired when continues to play.

• **VIDEO_STOPPED:** Fired when stopVideo is called.

• **VIDEO_CLEARED:** Fired when clearVideoPlayer is called.

• **VIDEO_LOOP COMPLETE:** Fired when all video loops have completed.

• **VIDEO_STARTED:** Fired when video meta data has loaded, this means the video is ready to play and will start unless paused at start is set.

• **UPDATE_VIDEO_ID:** Fired when the videoReportingID has changed, this happens when a new video is loaded.

• **VIDEO_BUFFER_EMPTY:** Fired when first starts.

• **VIDEO_BUFFER_FULL:** Fired when buffer is full.

• **SHOW_BUFFERING:** Fired when video is buffering.

• **HIDE_BUFFERING:** Fired when video is done buffering.

• **VIDEO_COMPLETE:** Fired when current video is done playing.

• **VIDEO_ERROR:** Fired when video failed for some reason.

• **VIDEO_BYTES_LOADED:** Fired as the video is being loaded, will pass a event.data.bytesLoaded value that is between 0 and 1.

• **VIDEO_PROGRESS:** Fired as video plays, passes a event.data.progress value that is between 0 and 1.

• **VIDEO_ON_CUEPOINT:** Fired when the video has reached a cue point that is embedded in the video. The cue point data object is passed through the event, cuepoint:Object = event.data.pointObj, this object will contain all the cue point data from the video.

• **VIDEO_0_PERCENT:** Fired when video first starts.

• **VIDEO_25_PERCENT:** Fired when video is at 25% progress.

• **VIDEO_50_PERCENT:** Fired when video is at 50% progress.

• **VIDEO_75_PERCENT:** Fired when video is at 75% progress.

• **VIDEO_100_PERCENT:** Fired when video is at 100% progress.

VIDEO TARGET MOVIE CLIP:

The video target movie clip is the display object that the video will be attached to. It should not be scaled but should contain a shape or graphic that is the size of the video. The player will get the video size from the target movie clip. It has to exist on the stage at the same time the video player is initialized. It can be nested in other movie clips. This movie clip will be passed through the video players constructor.

var vidPlayer:DCVideoPlayer = new DCVideoPlayer(**videoTarget_mc**);

VIDEO PLAYER CONSTRUCTOR VALUES:

You can pass several values through the constructor when you init the player. Only the video target is required, the rest can be left out, they have default values that will be used if no value is passed.

var vidPlayer:DCVideoPlayer = new DCVideoPlayer(**_videoTarget:MovieClip, _startMuted:Boolean = true, _useTunein:Boolean = false, _streaming:Boolean = false, _streamingPath:String = ""**);

_videoTarget:MovieClip
This is described above in **"VIDEO TARGET MOVIE CLIP".**

_startMuted:Boolean = true
This is a true false value, the default is to start the video muted, sound can be turned on after using either controls or setting the value to the player directly, **vidPlayer.unmuteAudio();** This will also update the sound control button if it is being used. If you want the video to start with audio on then pass false.

_useTunein:Boolean = false
This value is false by default. It tells the player to use date coded videos using the TuneIn class. If the videos aren't date coded or your using your own tune in logic then leave it as false. If your using the TuneIn class and the videos have a date coded value when passed to the player then set to true.

_streaming:Boolean = false
This value sets up the player to stream the video instead of a progressive download. This is actually optional for Media Mind, their player sniffs the video path and server connection and decides how best to handle it.

_streamingPath:String
This is optional for streaming, you can pass the full rtmp path with each video or pass a path to the video folder here.

VIDEO PLAYER PUBLIC PROPERTIES:

Once the player is defined you can set other properties that are listed below.

playerID:String = "default"
This is the players instance id, a string that defines just this player. Since the video events bubble, if you have multiple players you may need to filter out events from different players. If you set this value you can then filter by the player id.

```
vidPlayer.playerID = "mainPlayer";
function videoComplete( e:BltVideoEvent ):void
{
	// filtering by player id, if player is not "mainPlayer" it will get ignored.
	if( e.data.playerId != "mainPlayer" ) return;

	showVideoThumbs();
}
```

tuneInID:String
This is used for setting a custom tuneIn id, the default is "default", should never have to change it but here you go.

replayVideoNum:int
This value only works if you used the addVideo method to add videos to the player. If you have an init video that plays mute and need to call replay with sound but don't pass a value or use the mute/unmute control to turn the sound on it will replay the video specified by this value. The addVideo method adds the videos to an array based on date id, this value represents a position on that array.

ignoreControls:Boolean = false
This tells the player to ignore control events, if this is set to true the video controls will have no effect on the player. You can still call player controls directly, **vidPlayer.playVideo();** will still work. This is important only if you have multiple players on the stage at one time. Like a background video animation, if the value is not set to true and you have a video player with controls on at the same time it could update the video controls on the main player.

bufferTime:Number = 1
Buffer is set to 1 second by default, if you want to change it just pass it. Time is in seconds.
vidPlayer.bufferTime = 3;

closeOnComplete:Boolean = false
This tells the player to destroy itself and clean up after the video completes.

videoSmoothing:Boolean = true
Sets the video smoothing value, it's true by default, not sure why you would want it false but it's there if you need to change it.

loopVideoNum:int = -1
This sets a loop value for a video, it will loop the video based on this value. After the specified loops are completed it will fire the **VIDEO_LOOP_COMPLETE** event, the **VIDEO_COMPLETE** event will fire at the end of each loop.

VIDEO PLAYER PUBLIC METHODS:

init():void
Once the the player is initialized and all the properties have been set then this needs to be called, this tells the player that it is cleared to set it self up.

destroy():void
This method stops the video, clears the stream and video component along with clear it from memory. Once this is called the player will have to be completely re-initialized to be used again. This is also called when the video player target movie clip is removed from stage.

addVideo(path:String, reportingID:String = "video", day:String = "default"):void
This method is used to add videos to the videoListManager.

- **path:** path to the video, could be to a server or relative local path, also a streaming path minus the app path.

- **reportingID:** This value is for tracking events from the current video, all video events carry this value for the current video.

- **day:** The videos are added to an array that is defined by the day value. If use tuneIn was selected the day value would correspond to the tuneIn id's.

addProgressEvent(time:String, videoID:String, method:Function, useTime:Boolean = false):void
This method allows you to add callbacks (functions) that get called when a video hits a certain point that is defined by the time value. The videoID allows you to define what video calls what callbacks.

- **time:** a value between 0 and 1 that is a percent of the current video playback position. If you pass 0.5 then it will fire when the video is 50% completed, 0.75 will fire at 75% completed. You can also pass time, "21" is 21 seconds, "1:21" is a minute and 21 seconds. To use time instead of a percentage (0 to 1 value) the useTime value must be set to true.

- **videoID:** This is the videoReportingID that you pass using addVideo or loadAndPlay, this insures that these methods will only fire when that video is playing.

- **method:** this is the callback function, when the video reaches the time value it will fire this function, it will also pass the time value that was hit when the function was called. So you can use one callback function and just parse out the result by the time value or have multiple callbacks, one for each event, just need to make sure to allow for a return number in the function, 'callback (time:Number = 0)', if the return value is not include it will error.

- **useTime:** Set this to true in order to pass a time value instead of a percentage (0 to 1 value) set this value to true, it is false by default. You can pass time as, "21" which is 21 seconds, or "1:21" which is a minute and 21 seconds.

playInitVideo():void
This only works if videos have been added to the player using the addVideo method. This will play the first video in the video list of the current day.

loadNewVideo(vid:int, initLoad:Boolean = true):void
This method is also for use if videos were added using the addVideo method. When you call it you pass a value that defines which video to load from the current video list. The initLoad value defines if it's the first video which would be false, which only matters if video is starting muted.

loadAndPlay(path:String, reportingID:String = "default"):void
This method is for just adding a single video and having it play immediately. This is good if it's a background animation video, transition video, short video with no sound.

loadAndPause(path:String, reportingID:String = "default"):void
This method does the same as above accept the video waits for a command to play.

replayVideo():void
Replays current video, will not change sound value.

replayWithSound(num:int = -1):void
This method will replay the current video and turn the sound on, or you can pass a value to play a different video from the current video list, this will only work if videos were added to the player using the addVideo method.

playVideo():void
Plays or resumes the currently loaded video.

setVideoVolume(_vol:Number):void
This allows you to change the video volume, takes a value between 0 and 1, 0.5 would be 50%.

muteAudio():void
Mutes video.

unmuteAudio():void
Unmutes video.

pauseVideo():void
Pauses currently playing video, not a toggle, if video is already paused it will be ignored.

stopVideo():void
This actually closes the current video stream, use only when done with the video.

seekVideo(e:BltVideoEvent, _seek:Number = 0):void
Normally this is called from the seek/progress bar component. If you wish to call it directly then pass null for the event and pass a value between 0 and 1 for the seek: **seekVideo(null, .5),** that will seek the video to 50%.

resetVideo():void
This will pause the video and send it back to the first frame.

skipVideo(_skip:Number = .9):void
This is for testing, this when called will skip the currently playing video to a specified value, the default is 90% of it's current length.

clearVideoPlayer():void
This will completely clean up the video player, removes it from the stage, this is also called when you call destroy.

VIDEO PLAYER SET UP SINGLE VIDEO, STARTS MUTED:

```
import com.blt.video.doubleClick.DCVideoPlayer;

var videoPlayer_mc:MovieClip = path to video target movie clip

var vidPlayer:DCVideoPlayer = new DCVideoPlayer( videoPlayer_mc );
vidPlayer.ignoreControls = true;

vidPlayer.init();
vidPlayer.loadAndPlay( "video1.flv", "video1" );
```

VIDEO PLAYER SET UP SINGLE VIDEO WITH EVENTS, STARTS MUTED:

```
import com.blt.video.doubleClick.DCVideoPlayer;
import com.blt.events.BltVideoEvent;

var videoPlayer_mc:MovieClip = path to video target movie clip

var vidPlayer:DCVideoPlayer = new DCVideoPlayer( videoPlayer_mc );
vidPlayer.addEventListener( BltVideoEvent.VIDEO_COMPLETE, videoComplete );

vidPlayer.init();
vidPlayer.loadAndPlay( "video1.flv", "video1" );


function videoComplete( e:BltVideoEvent ):void
{
        trace( "VIDEO COMPLETE" );
        // clean up video player
        vidPlayer.destroy();
        vidPlayer = null;

        gotoAndPlay( "endframe" );
}
```

VIDEO PLAYER SET UP MULTIPLE VIDEOS, NOT DATE CODED, STARTS MUTED:

```
import com.blt.video.doubleClick.DCVideoPlayer;


var vidPlayer:DCVideoPlayer;
var videoTarget:MovieClip = videoComponent_mc.videoPlayer_mc;

vidPlayer = new DCVideoPlayer( videoTarget );

vidPlayer.addVideo( "video1.flv", "video1" );
vidPlayer.addVideo( "video2.flv", "video2" );
vidPlayer.addVideo( "video3.flv", "video3" );
vidPlayer.addVideo( "video4.flv", "video4" );
vidPlayer.init();
vidPlayer.playInitVideo();
```

VIDEO PLAYER SET UP DATE CODED VIDEOS, DOES NOT START MUTED:

```
import com.blt.video.doubleClick.DCVideoPlayer;


var vidPlayer:DCVideoPlayer;
var videoTarget:MovieClip = videoComponent_mc.videoPlayer_mc;

vidPlayer = new DCVideoPlayer( videoTarget, false, true );

// tonight videos
vidPlayer.addVideo( "video1.flv",        "video1", "tonight" );
vidPlayer.addVideo( "video2.flv",        "video2", "tonight" );

// tomorrow videos
vidPlayer.addVideo( "video1.flv",        "video1", "tomorrow" );
vidPlayer.addVideo( "video2.flv",        "video2", "tomorrow" );

// sunday videos
vidPlayer.addVideo( "video1.flv",        "video1", "sunday" );
vidPlayer.addVideo( "video2.flv",        "video2", "sunday" );

vidPlayer.init();
vidPlayer.playInitVideo();
```