

# Sonettic-Cinema

## **Skin Development Guide**

© 2009 Sonettic  
[www.sonettic-cinema.com](http://www.sonettic-cinema.com)

# Sonettic Cinema Player Skin Development Guide

## Skin Development Overview

All Sonettic Cinema Kits contain Development Version of Sonettic Cinema Player. Sonettic Cinema Dev. Player is fully Sonettic Cinema HD Player without any included skin. Also in kit included CinemaAqua skin source (Adobe CS3 FLA and AS3 AS files and examples.

To create CinemaAqua Skin i spent only one day. You have this skin source and you have ability to change all skin graphics. But if you want to understand Cinema Player and wish to have more developer power you should know more about Sonettic Cinema Player Development model.

## Skin Controller Overview

*Class name: **MainUIControlsComander***

*Package: **com.sonettic.GUI***

MainUIControlsComander does connect skin and set initialized data. Skin might get all necessary data from MainUIControlsComander. For more details discover this class.

## Cinema Player Event Model overview and exercise

Sonettic Cinema HD Player built on Message Model. Cinema Player have one main Event Connector. This connector instance set to all modules of player. Modules uses MediaConnector to send and receive messages.

Event Connector is event broadcaster.

*Class name: **MediaConnector***

*Package: **com.sonettic.connector***

MediaConnector broadcast events of MediaEvent type:

*Class name: **MediaEvent***

*Package: **com.sonettic.media***

MediaEvent types list:

*Class name: **ConnectorEventsList***

*Package: **com.sonettic.connector***

All about MediaEvent object and Event Type List you can get from [Player Events PDF](#) .

## Exercise

Create Button of external Skin ( for internal development typed all properties, received and returned values )

```
package com.sonettic.GUI{

    import flash.display.Sprite;
    import flash.events.*;

    class MyPlayPauseButton extends Sprite{

        private var Connector:*;
        private var BValue:uint;
        private var EventName:String;

        function MyPlayPauseButton(){
            value = 0;
        }

        /**
         * Set Media Event Connector
         * @return mconnector:MediaConnector
         *
         * @see com.sinema.cconnector.MediaConnector
         * @see com.sonettic.media.MediaEvent
         * @see com.sonettic.connector.ConnectorEventsList
         */
        public function get connector():* {
            return Connector;
        }
        /**
         * @private
         */
        public function set connector(mconnector:*):void {
            Connector=mconnector;
        }

        /**
         * Get Button Value
         * @return uint
         */
        public function get value():uint{
            return BValue;
        }
        /**
         * @private
         */
        public function set value (val:uint):void{
            BValue = val;

            if(value == 0 ){
                EventName = "uiPlayClick";
            }
            if(value == 1 ){
                EventName ="uiPauseClick";
            }
        }

        // init all event listeners
        private function initEventListeners():void{
            Connector.addEventListener("mediaPlay", mediaPlayListener);
            Connector.addEventListener("mediaPause", mediaPauseListener);
        }
    }
}
```

```

Connector.addListener("mediaStop", mediaStopListener);
Connector.addListener("mediaComplete", mediaStopListener);

// click on button
this.addListener(MouseEvent.CLICK, mouseClickListener);

}

// mouse event listener
private function mouseClickListener(e:MouseEvent):void{
    // broadcast Event
    eventDispatch(EventName, this);
}

// event listeners
private function mediaPlayListener(e:MediaEvent):void{
    value=1;
}
private function mediaPauseListener(e:MediaEvent):void{
    value=0;
}
private function mediaStopListener(e:MediaEvent):void{
    value=0;
}

//-----
//      EVENT PORT
//
// Create and transfer Object to Media Connector. Media Connector broadcast it as MediaEvent
//-----
/**
 * eventDispatch() - method create and transfer Object to Media Connector.
 *                   Media Connector broadcast it as MediaEvent
 * @param type:String - Event Type
 * @param target:~~ - event owner
 * @param data:~~ @default null
 *
 * @see com.sinema.cconnector.MediaConnector
 * @see com.sonettic.media.MediaEvent
 * @see com.sonettic.connector.ConnectorEventsList
 */
protected function eventDispatch(type:String, target:*, data:*=null):void {
    var eventObject:Object=new Object();
    eventObject.type=type;
    eventObject.target=target;
    eventObject.value=data;
    if (Connector) {
        Connector.eventsPort(eventObject);
    }
}
}

var MyPPButton:MyPlayPauseButton = new MyPlayPauseButton();

// set media connector to button ( Connector is MediaConnector and initialized earlier)
MyPPButton.connector = Connector;

// add button to some container
addChild(MyPPButton);

```

## CinemaAqua external Skin overview

All Kits contain CinemaAqua external skin source files. CinemaAqua Class is main skin Class.

*Class name:* **CinemaAqua**  
*Package:* **com.sonettic\_cinema.skins.aquastandard**

SntMainUIControls is skin Components Controller. Skin Components Controller must be implemented ISkin interface.

*Class name:* **SntMainUIControls**  
*Package:* **com.sonettic\_cinema.skins.aquastandard.controlsui**

SntMediaMenu is visual skin component player menu.

*Class name:* **SntMediaMenu**  
*Package:* **com.sonettic\_cinema.skins.aquastandard.controlsui**

## Advanced Development

### Add new button to Player Submenu

Get Player SubMenu:

```
private var ModuleName:String = "MyModuleName";
private var SubMenu:*;
private var BubbleController:*;

//-----
// Get Objects from Player Objects Repository
//-----
private function getObjectsFromRepository():void{

    // Method get objectRepository
    // Obhets Repository contain alot modules and objects and this module too
    // get Object from repository by Object Name
    var MObjetRepository:* = ModuleController.objectRepository;

    // get player Submenu from ObjectsRepository
    //com.sonettic.utilities.ObjectsRepository
    SubMenu = MObjetRepository.getObject("SUBMENU");
    if(SubMenu == null){
        // wait when object will be added to ObjectRepository
        // add event objectSet_ + object name
        // remove event objectDelete_ + object name
        // all objects removed objectClear
        MObjetRepository.addEventListener("objectSet_SUBMENU",
                                        submenuAddedObjectRepositoryEventListener);
    }

    //-----
    // SubMenu added to ObjectsRepository Event Listener
    //-----
    private function submenuAddedObjectRepositoryEventListener(e:Event):void{
        // create module icon and set it to player Submenu
    }
}
```

```
        createModuleIcon();  
    }  
}
```

To add new Button in player standard submenu we have to create icons graphics contained in Sprite.

If button should have two states, we should to create two icons graphics contained in Sprite.

Icon sizes 64X64

1 – state **ON** ( active )

2 – state **OFF** ( passive )

## Go to create button

```
//-----  
// CREATE MODULE ICON IN PLAYER SUBMENU  
//-----  
private function createModuleIcon():void{  
    // if submenu is null  
    if(!SubMenu ) return;  
    //figure out button pattern at com.sonettic.GUI.icons.IconPattern;  
  
    // UIControlComander is skin controller of player  
    var SharedIcons:* = UIControlComander.getIconsLibrary();  
    // trace Shared Library Components List  
  
    // Shared Library contain buttons and components  
    trace(SharedIcons.getIconsList);  
  
    // get Play Button from Shared Library  
    var PlayIcon:*=SharedIcons.getIcon("Play");  
  
    // we should get ButtonPatter to create new Button  
    var ModuleIconInstance:* = SharedIcons.getIcon("ButtonPattern");  
  
    // set active icon graphic  
    ModuleIconInstance.activeicon = new ModuleActiveIcon();  
  
    // set passive icon graphic  
    ModuleIconInstance.passiveicon = new ModulePassiveIcon();  
  
    // set Icon Events names  
    // click event  
    ModuleIconInstance.eventName = ModuleName+"_uiIconClick";  
  
    // RollOver event  
    ModuleIconInstance.eventRollOverName = ModuleName+"_uiIconOver";  
  
    // RollOut event  
    ModuleIconInstance.eventRollOverName = ModuleName+"_uiIconOver";  
  
    // DoubleClick event  
    ModuleIconInstance.eventDoubleClickName = ModuleName+"_uiIconDoubleClick";  
  
    // set double click enable  
    // it will be set to true if boubleClickEventName set  
    //ModuleIconInstance.doubleClickEnable = false;  
  
    // Button Hilt
```

```
ModuleIconInstance.hint="Module Icon";

// set hint from language object
//ModuleIconInstance.hintName = language object field or language XML node name

// add icon to player submenu
//addItem(item:*, sortindex:int=100, uniqueitem:Boolean=true)
SubMenu.createItem(ModuleIconInstance, 0);
}
```

## Create new Bubble Control and add it to Player Bubble Controller

Bubble Control is some components added to Bubble Container through Bubble Controller.

*Class Name:* **BubbleController**  
*Package:* **com.sonettic.GUI.bubblecontrols**

To create own new Bubble we should use BubblePattern:

*Class Name:* **BubblePattern**  
*Package:* **com.sonettic.GUI.bubblecontrols**

```
//-----
// Get Objects from Player Objects Repository
//-----
private function getObjectsFromRepository():void{

    // Method get objectRepository
    // Obhets Repository contain alot modules and objects and this module too
    // get Object from repository by Object Name
    var MObjetRepository:* = ModuleController.objectRepository;

    //get BubbleController from ObjectsRepository
    //BubbleController com.sonettic.GUI.bubblecontrols.BubbleController
    //ObjectsRepository com.sonettic.utilities.ObjectsRepository
    BubbleController = MObjetRepository.getObject( "BUBBLECONTROLLER" );

    // create bubble control
    createModuleBubble();
}

//-----
// Get Module Bubble Control
//-----
private function createModuleBubble():void{
    if(BubbleController == null) return;
    // get Bubble Pattern from Player Bubble Controller
    var MBubble:* = BubbleController.getBubblePattern();
    // type 1 without bubble icon
    // type 2 with bubble icon
    MBubble.initPattern(1);
    // set icon if bubble is type 2
    MBubble.setIcon(new ModuleActiveIcon());
    // set bubble name
    MBubble.iname = ModuleName+"Bubble";
    // set bubble content
    MBubble.content= new BubbleContent();
}
```

```
// show bubble when event occurred
MBubble.showEvent = ModuleName+"_uiIconClick";
// add bubble to BubbleController
BubbleController.addBubble = MBubble;
}
```

## Connect Skin as External

Connect external skin through:

1. **flashvars**
2. **internal properties** in **MainProperties.as**
3. **external XML properties** file

properties is skin=http://mysite.com/myskin.swf

## Add Skin as Internal

Create Skin with unique namespace

and connect it through MainUIControls Instance in **com.sonettic.main**

```
//-----
private function connectPlayerGUI():void{
    //-----
    // set complete callback
    //-----
    MainUIControls.completeCallback = skinCreateCompleteEventListener;

    //-----
    //      set Current SKIN
    //      com.sonettic.GUI.skins.cinemaglow.CinemaGlow
    //-----
    MainUIControls.skin = new MySkin();

    //-----
    // set path to External Skin (SWF )
    //-----
    MainUIControls.externalskin=GlobalSettings.skin;

    //-----
    // listener skin connected
    //-----
    function skinCreateCompleteEventListener():void{
        //-----
        //          LOAD EXTERNAL MODULES
        //-----
        loadExternalModules(GlobalSettings);
    }
}
```