Prerelease Notes

Flash Player 11.7 and AIR 3.7 Release Notes

Welcome to Adobe® Flash® Player 11.7 and Adobe AIR 3.7!

Last Updated: April 30, 2013

This pre-release includes new features as well as enhancements and bug fixes related to security, stability, performance, and device compatibility for Flash Player 11.7 and AIR 3.7. This document may be updated periodically as more information becomes available.

NOTE:

- The ActiveX Flash Player in this release is not compatible with Windows® 8
  .Flash Player for Windows® 8 is available as part of the generally available Windows® 8 update

Release features

- **Sandboxing enhancements**
  This feature adds enhancements to the existing sandbox feature by better addressing application launches in protected mode
- **Prevent Cloud backup for Shared Objects (iOS)**
  Currently in AIR applications for iOS, there is no way for developer to prevent the local shared object data from being backup on Apple iCloud. With AIR 3.7, a new static property, called 'preventBackup' has been introduced to SharedObject ActionScript class which will enable developers to address this limitation. This property can be used only with 3.7 or greater namespaces (swf-version=>20). Setting this property to true will prevent all the shared objects from being backed up. The default value of the property is false, this property will work only on devices having iOS 5.1 or higher.

```
var mySO:SharedObject;
mySO = SharedObject.getLocal("test");
mySO.data.userName = "developer";
mySO.data.itemNumbers = new Array(1,2,3);
SharedObject.preventBackup = true;
mySO.flush();
```

- **Use CPU Render Mode For selected devices while using GPU on others (iOS)**
  A new tag, <forceCPURenderModeForDevices> has been added for iOS devices and it would be applicable from AIR 3.7 (swf-version = 20). This tag would force CPU render mode for a given set of iOS devices and have gpu renderMode for all the remaining iOS devices. This feature could be used, when using gpu render mode results in insufficient memory on some of the low end devices (like iPad1 and iPod4).
  e.g. If we specify <renderMode> GPU </renderMode> and <forceCPURenderModeForDevices> iPhone1 iPad1 </forceCPURenderModeForDevices>
  then all iOS devices other than iPad1 and iPhone1 will have renderMode GPU. These devices will have CPU as render mode.
  Please note that this tag currently applies to iOS only. The tag needs to be added to the platform specific section under the iPhone tag.

```
<iPhone>
    <InfoAdditions>
      <![CDATA[
         <key>UIDeviceFamily</key>
         <array>
           <string>1</string>
           <string>2</string>
         </array>
         <key>UIStatusBarStyle</key>
         <string>UIStatusBarStyleBlackOpaque</string>
         <key>UIRequiresPersistentWiFi</key>
         <string>YES</string>
      ]]>
    </InfoAdditions>
    <forceCPURenderModeForDevices> iPhone3,1 iPad </forceCPURenderModeForDevices>
</iPhone>
```

Tag <forceCPURenderModeForDevices> has been added under the iPhone tag, and it accepts a space separated list of device model names. Some of the valid device model names are listed below.

```
"iPod4,1"   // iPod Touch Fourth Generation
"iPod5,1"   // iPod Touch Fifth Generation
"iPhone2,1" // iPhone 3GS
"iPhone3,1" // iPhone 4
"iPhone3,2" // iPhone 4 CDMA
"iPhone4,1" // iPhone 4S
"iPhone5,1" // iPhone 5
"iPad1,1"   // iPad
"iPad2,1"   // iPad 2
"iPad2,2"   // iPad 2 (GSM)
"iPad2,3"   // iPad 3 (CDMA)
"iPad2,4"   // iPad 3 (CDMAS)
"iPad2,5"   // iPad Mini Wifi
"iPad3,1"   // iPad 3 (WIFI)
"iPad3,2"   // iPad 3 (CDMA)
"iPad3,3"   // iPad 3 GSM
"iPad3,4"   // iPad 4 (Wifi)
```

- **External hosting of secondary swf files (iOS)**
  Application developers will be able to host their secondary SWFs on an External server and load them on demand as per their application logic using this feature. The loading secondary SWFs which have any ABC code in AOT mode, which worked for just locally packaged SWFs earlier, will now work for loading SWFs externally as well.
  The developer would need to change the URL of Loader's URLRequest to the URL of his externally hosted stripped SWF. A sample URL request for using this feature is

```
private var externalSwfUrl:String= "http://www.xyz.com/ExternalSwf.swf";
private var urlRequest:URLRequest = new URLRequest(externalSwfUrl);
```

To enable this feature, developer will have to specify a text file containing details of the SWF files to be stripped & externally hosted. The developer needs to specify line separated paths of SWFs which are to be externally hosted within this text file. The format of specifying SWF Files within a Sample .txt file is as follows :

```
assets/Level1.swf
assets/Level2.swf
assets/Level3/asset/Level3.swf
```

This text file needs to be mentioned in the <externalSwfs> Tag under the <iPhone> Tag within the application descriptor as shown below :

```
<iPhone>
       .
       .
       <externalSwfs>assets/SampleSWFInfoFile.txt</externalSwfs>
       .
       .
</iPhone>
```

During ADT packaging, the developer needs to specify the text file just like an asset along with the set of SWF's specified in the text file. A sample ADT command for using this feature is:

```
adt -package -target ipa-app-store -provisioning-profile <Provisioning Profile> -storetype
pkcs12 -keystore <Certificate> -storepass <Password> ResultantIPA.ipa SampleMainSwf-app.xml
SampleMainSwf.swf assets/SampleSWFInfoFile.txt assets/Level1.swf assets/Level2.swf
assets/Level3/asset/Level3.swf
```

During IPA packaging, ADT extracts the Actioscript code from all child SWFs which are specified within the sample text file, adds it to the final executable and moves the stripped SWFs into the "externalStrippedSwfs" folder created in the current working directory. The directory structure within the "externalStrippedSwfs" folder remains the same as specified within the text file. The generated stripped SWF's can be externally hosted on a web server of developer's choice.
A working sample Actionscript code which loads a secondary swf derived using the above workflow, from an external server is as follows :

```
package
{
 import flash.display.Loader;
 import flash.display.Sprite;
 import flash.events.Event;
 import flash.events.IOErrorEvent;
 import flash.net.URLRequest;
 import flash.system.ApplicationDomain;
 import flash.system.LoaderContext;

 public class SampleMainSwf extends Sprite
 {
  private var externalLoader:Loader;
  private var url:String= "http://www.xyz.com/Level1.swf";
  private var urlRequest:URLRequest = new URLRequest(url);
  private var ldrContext:LoaderContext;

  public function SampleMainSwf()
  {
   externalLoader = new Loader();
   externalLoader.contentLoaderInfo.addEventListener(Event.COMPLETE,completeHandler);
   externalLoader.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR,errorHandler);
   ldrContext=new LoaderContext(false,ApplicationDomain.currentDomain,null);

   try
   {
    externalLoader.load(urlRequest, ldrContext);
   }
   catch(e:Error)
   {
    trace("Error ID : "+e.errorID+"\nError Message : "+e.message);
   }
  }

  private function completeHandler(e:Event):void
  {
   addChild(externalLoader);
  }

  private function errorHandler(e:IOErrorEvent):void
  {
   trace("In IO ErrorEvent Handler");
  }

 }
}
```

Note:
1. The current releases of Flash Builder 4.7/Flash Professional CS6 do not support this feature. To enable generation of Stripped SWF for external hosting, command line ADT packaging should be used.
2. The loading of Flex secondary SWFs causes the application to crash.

- **GameInput API (Android)**
  The GameInput API is an interface that allows applications to communicate with attached game input devices (joysticks, gamepads, wands, etc). There is a wide range of game input devices, which have different capabilities and controls. This API is only supported on Android devices running on OS version 4.1 or higher. This API is implemented such that it might function well with different (and possibly unknown) types of game input devices.
  This feature can only be used with namespace 3.7 or greater and requires the minimum swf version to be equal to 20.

```
private var gameInput:GameInput;
    public function TestGameInput()
    {
        trace("GameInput.isSupported - "+ GameInput.isSupported);
        trace("GameInput.numDevices - "+ GameInput.numDevices);

        gameInput = newGameInput();
        gameInput.addEventListener(GameInputEvent.DEVICE_ADDED, inputDeviceAddedEvent);
        gameInput.addEventListener(GameInputEvent.DEVICE_REMOVED, inputDeviceRemovedEvent);

        function inputDeviceRemovedEvent(e:GameInputEvent):void
        {
            trace("inputDeviceRemovedEvent - "+ e.device);
        }

        function inputDeviceAddedEvent(e:GameInputEvent):void
        {
            trace("inputDeviceAddedEvent - "+ e.device);
            getDeviceInformation(e.device);
        }

        function getDeviceInformation(device:GameInputDevice):void
        {
            trace("device.enabled - "+ device.enabled);
            trace("device.id - "+ device.id);
            trace("device.name - "+ device.name);
            trace("device.numControls - "+ device.numControls);
            trace("device.sampleInterval - "+ device.sampleInterval);

            for(var i:Number=0; i < device.numControls; i++)
            {
                var control:GameInputControl = device.getControlAt(i);
                getControlInformation(control);
                control.addEventListener(Event.CHANGE, changeEvent);
            }
        }

        function changeEvent(e:Event):void
        {
            var control:GameInputControl = e.target as GameInputControl;
            getControlInformation(control);
        }

        function getControlInformation(control:GameInputControl):void
        {
            trace("control.device - "+ control.device);
            trace("control.value - "+ control.value);
            trace("control.minValue - "+ control.minValue);
            trace("control.maxValue - "+ control.maxValue);
            trace("control.id - "+ control.id);

        }
    }
```

- **Packaging applications with only captive runtime (Android)**
  With AIR 3.7, packaging of AIR applications for Android in any target will embed the AIR runtime. This would help in improving the user experience as there would be no need to download the AIR runtime separately. A side-effect would however be that there would be an increase in app size of around 9MB.

Builds changes

Runtime Versions

Flash Player Desktop only: **11.7.700.197**
AIR Runtime Desktop: **3.7.0.1820**
AIR Runtime Android: **3.7.0.1820**
AIR SDK: **3.7.0.1820**

Authoring

Authoring for Flash Player 11.7

To use the new Flash Player, you will need to target SWF version 20 by passing in an extra compiler argument to the ASC 2.0 compiler: -swf-version=20.Directions are below:

- Download the new playerglobal.swc for Flash Player 11.7
- Download and install Flash Builder 4.7 from Creative Cloud: -https://www.adobe.com/products/gaming/tools.html
- Backup the existing AIR SDK if you need to restore it later then replace the bundled AIR SDK with the AIR 3.7 SDK. To do this, unzip the AIR 3.7 SDK to this location:\
    - MacOS--: /Applications/Adobe Flash Builder 4.7/eclipse/plugins/com.adobe.flash.compiler_4.7.0.348297/AIRSDK
    - Windows: C:\Program Files\Adobe\Adobe Flash Builder 4.7 (64 Bit)\eclipse\plugins\com.adobe.flash.compiler_4.7.0.349722\AIRSDK\
- In Flash Builder, create a new project: File -> New -> project.\
- Open the project Properties panel (right-click and chose 'Properties'). Select\ ActionScriptCompiler from the list on the left.
- Add to the 'Additional compiler arguments' input: \swf-version=20. This ensures the outputted SWF targets SWF version 20. If you compile on the command-line and not in Flash Builder, you need to add the same compiler argument.

Authoring for AIR 3.7 Update to the AIR 3.7 namespace

You must update your application descriptor file to the 3.7 namespace in order to access the new AIR 3.7 API's and behavior. If your application does not require the new AIR 3.7 API's and behavior, you are not required to update the namespace. However, we recommend all users start using the AIR 3.7 namespace even if you are not yet taking advantage of the new 3.7 capabilities. To update the namespace, change the xmlns attribute in your application descriptor to: <application xmlns="http://ns.adobe.com/air/application/3.7">

System Requirements

For current Flash Player system requirements visit- http://www.adobe.com/products/flashplayer/systemreqs/
For current AIR system requirements, visit -http://www.adobe.com/products/air/systemreqs/

Flash Player 11.7 has the following minimum system requirements:

-	Windows	Macintosh
Processor	2.33 Ghz or faster x86-compatible processor, or Intel® Atom™ 1.6GHz or faster processor for netbook class devices{-}	Intel® Core™ Duo 1.83GHz or faster processor
Operating System	Microsoft® Windows® XP (32-bit), Windows Server 2008 (32-bit), Windows Vista® (32-bit), Windows 7 (32-bit and 64-bit)	Mac OS® X 10.6, Mac OS X 10.7, Mac OS X 10.8
Browser	Internet Explorer 7.0 and above, Mozilla Firefox 17.0 and above, Google Chrome, Safari 5.0 and above, Opera 11	Safari 5.0 and above, Mozilla Firefox 17.0 and above, Google Chrome, Opera 11
Memory	512MB of RAM (1GB RAM recommended for netbook class devices), 512MB of graphics memory	256MB of RAM, 512MB of graphics memory

AIR 3.7 has the following minimum system requirements:

-	Windows	Macintosh	Android	iOS
Processor / Device Hardware	2.33GHz or faster x86-compatible processor or Intel Atom™ 1.6GHz or faster processor for netbook class devices	Intel® Core™ Duo 1.83GHz or faster processor	ARMv7 processor with Vector FPU, Minimum 550MHz, ES2.0, H.264 & AAC H/W Decoders	iPod touch 4, iPod touch 5, iPhone 3GS, iPhone 4/4S, iPad, iPad 2, iPhone 5, The New iPad, iPad mini
Operating System	Microsoft® Windows® XP, Windows Server® 2003, Windows Server® 2008, Windows Vista® Home Premium, Business, Ultimate, or Enterprise (including 64-bit editions) with Service Pack 2, or Windows 7{-}	Mac OS® X 10.6 and 10.7	Android 2.3 and above	iOS 4.3 and higher
RAM	512MB of RAM (1GB recommended)	512MB of RAM (1GB recommended)	256MB RAM	