Welcome to Adobe® Flash® Player 23 and Adobe AIR 23!

**Last Updated: October 19, 2016**

Welcome to the latest Flash Runtime version 23 beta! We've been hard at work adding new features to Flash Player and AIR and are looking forward to receive feedback from our development and Flash Player community.

This beta release includes new features as well as enhancements and bug fixes related to security, stability, performance, and device compatibility for Flash Player 23 and AIR 23. This document may be updated periodically as more information becomes available.

As always, we appreciate all feedback. We encourage you to post in our beta forums or create bug reports or feature requests on our public bug database.

**Flash Player Beta Forum**  
**AIR Beta Forum**  
**Bug Database**

**NOTE:**

- The ActiveX Flash Player in this release is not compatible with Windows® 8.x or 10  
- Flash Player for Windows® 8.x/10 is available as part of the generally available Windows® 8.x/10 update

**New and Updated Features**

**Extended Beta - Flash Player NPAPI for Linux**

Linux users have access to both NPAPI and PPAPI versions of Flash Player. However, for the last four years, the NPAPI version has been held at 11.2 and regularly updated with only security fixes while the PPAPI version (used in Chrome and Chromium based browsers), is in line with the standard Windows and Mac releases.

Today we are updating the beta channel with Linux NPAPI Flash Player by moving it forward and in sync with the modern release branch (currently version 23). We have done this significant change to improve security and provide additional mitigation to the Linux community.

In the past, we communicated that NPAPI Linux releases would stop in 2017. However, this is no longer the case, and once we have performed sufficient testing and received community feedback, we will release both NPAPI and PPAPI Linux with their major version numbers in sync and on a regular basis.

Because this change is primarily a security initiative, some features (like GPU 3D acceleration and premium video DRM) will not be fully implemented.

If you require this functionality, we recommend that you use the PPAPI version of Flash Player. That said, we believe that the new NPAPI build represents a significant step forward in functionality, stability, and security and look forward to hearing your feedback.

*Currently we are distributing 32 and 64 bit binaries only, we will package these in proper installers after testing and feedback. Users should manually back-up their existing Flash Player plugin file from the appropriate system plug-in folder and copy the new binaries into place to use them.*
GameInput API for iOS

Starting AIR 23, we are providing GameInput API support for iOS.

GameInput API is an interface that allows applications to communicate with attached Game controllers. There are different types of Game Controllers depending on the number of controls. This API allows an application to get all information related to the Game controller.

GameInput API support has already been available for Android. On iOS, it requires a minimum swf version of 34. This API supports iOS devices running on iOS version 9.0 or higher. Currently, this API supports extended Gamepads only.

GameInput API Example:

```ActionScript
private var gameInput:GameInput;

public function TestGameInput()
{
    trace("GameInput.isSupported - "+ GameInput.isSupported);
    trace("GameInput.numDevices - "+ GameInput.numDevices);

    gameInput = new GameInput();
    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);
    }
}
```

Echo Cancellation on AIR for Android

Starting AIR 23, we have introduced Echo Cancellation for AIR on Android.

Acoustic echo occurs when the speaker output feeds back to the built-in microphone input, producing disturbing echoing artifacts and significantly
reducing the quality of captured data. Acoustic echo arises in collaboration applications, where two or more parties carry out interactive conversation.

From this release onwards, developers can develop VOIP applications without any echo using AIR on Android.

**Note:** This feature is already present on AIR for desktop and iOS Platform. For more information, see Microphone - AS3.

The following changes are required in the action script code:

**Microphone.getEnhancedMicrophone**

To get access to device Microphone, we use Microphone.getMicrophone(). However, this API returns a simple microphone, which does not have the ability to eliminate acoustic echo. To remove the acoustic echo, developers must get an instance of Microphone using the API: Microphone.getEnhancedMicrophone(). The device microphone returned by this API has the acoustic echo cancellation feature enabled.

**Requirements**

- Add the following tag under Android manifest additions:

```xml
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
```

- There may be pre-existing swfs that use Microphone.getEnhancedMicrophone() API (as it is already present and working for AIR desktop applications). If such swf files are packaged with the latest AIRSDK (version 23), the feature will not work. Developers need to recompile the swf with swf-version 34 or higher.

**Sample snippet**

Here is the example code snippet for this scenario:

```actionscript
public function Microphone()
{
    mic = Microphone.getEnhancedMicrophone();
    mic.gain=60;
    mic.rate =22;
    mic.addEventListener(StatusEvent.STATUS, mic_status);
}
```

**Limitations**

- Acoustic Echo Cancellation may not work on some devices (such as the Moto G2) because of hardware side limitations.
- The API Microphone.getEnhancedOptions (present on AIR for desktop) will be a no-op on AIR for Android.
- Performance of the feature can vary depending upon the hardware side handling of echo cancellation for different Android devices. For the devices that do not support Echo Cancellation at the hardware level, AIR will handle the echo cancellation from the software side.
- Acoustic effects of the microphone class will vary according to the device. Because the Android Family has devices with different hardware configurations, the same audio settings will have different impact on different devices. For example, a developer may have to use "mic.gain = 70;" for Samsung Note 4 for loud output of the voice.

**The StageText clear button is now optional on iOS**

While creating StageText object on iOS, a clear button option is available by default to the AS developers.

Beginning with AIR 23, a new feature has been introduced, which makes the StageText clear button optional. This provides more flexibility to AS developers when using StageText. In earlier versions, StageText object was created with an integrated clear button.

To implement this feature, the following changes are required in the ActionScript code:

**stageText.clearButtonMode**

To get access to the optional clear button support, AS developers can now use a clearButtonMode property to set different modes. The following modes are associated with this property:

- To show clearButton while editing: StageTextClearButtonMode.WHILE_EDITING
- To never show clearButton: StageTextClearButtonMode.NEVER
- To always show clearButton: StageTextClearButtonMode.ALWAYS
- To show clearButton unless editing: StageTextClearButtonMode.UNLESS_EDITING

By default, clearButtonMode property is set as StageTextClearButtonMode.WHILE_EDITING.
Requirements

- The application must be packaged with the latest AIR SDK (version 23).
- SWF version must be 34 or later.

Sample snippet

Here is the example code snippet for this scenario:

```javascript
public function ClearButtonMode()
{
    label = new StageText();
    label.clearButtonMode = StageTextClearButtonMode.NEVER;
    label.textAlign = "right";
    label.stage = stage;
    label.viewPort = new Rectangle(20, 20, 90, 90);
}
```

Limitations

- This property is not available in StageText with multi-line
- This property is a No-Op for Android

HSTS Support in Flash Player

Beginning with Flash Player 23, we have introduced support for HSTS (HTTP Strict Transport Security). HSTS is an IETF standard, which enforces user agents (browsers) to use HTTPS for communication instead of HTTP. HTTPS response may have a Strict-Transport-Security (STS) header field that requests the user agent to make further requests in HTTPS. Flash Player will now acknowledge the STS header in HTTPS response.

This is particularly helpful when a SWF calls another SWF (child SWF) that is present in HSTS enabled server. Flash Player will acknowledge the STS header in the response and further request that the same domain will always be HTTPS. This feature is helpful in mitigating protocol hijacking attacks and cookie hijacking.

Disabling local-with-filesystem access in Flash Player by default

Beginning with Flash Player 23, local-with-network permissions will now be applied to all local SWF content, regardless of the preference chosen at compile time.

Background

When playing Flash (SWF) content from local filesystem, developers have historically been able to configure content to exclusively read from the filesystem, or communicate to the network. When this functionality was introduced over a decade ago, it enabled an interesting array of use-cases ranging from simple games to interactive kiosks. In context of modern web security, it is time to retire local filesystem functionality in the browser plugin. At the same time, Adobe AIR has been established as a robust, mature solution for delivering ActionScript-based content as a standalone application.

Vast majority of Flash Player users and content will be unaffected by this change. This change only impacts Flash content played from the local filesystem, using the browser. Flash content hosted on the internet and local webservers, as well as the Standalone Flash Player remains unaffected.

If you are a user who requires this functionality, these files can be added to the list of Trusted Locations in Flash Player.

Workarounds for Legacy Content

We highly recommend that you only circumvent these controls to enable content from sources that they trust.

For Individuals:

For Internet Explorer, Edge, Firefox, Opera and Safari:

- On the affected system, go to: Control Panel > Flash Player > Advanced > Developer Tools > Trusted Location Settings
- Click the (+) icon and add relevant SWF(s) to the list

For Google Chrome (and similar PPAPI browsers):

Choose Edit Locations > Add Locations from the list.

For System Administrators:
The legacy behavior can be restored by applying the EnableInsecureLocalWithFileSystem=1 flag to mms.cfg.

Video and Camera support for Stage3D by VideoTexture for Flash Player (Release)

In Flash Player 20 or earlier, use of video in Stage3D required use of the Video object, which is not hardware accelerated. It involved copying the video frame to a BitmapData object and then loading data onto the GPU, which made it CPU-intensive.

To address this limitation, Video texture object was introduced. It allows you to use hardware decoded video in Stage 3D content. This capability is further extended in Flash Player 23 release, and texture objects have been introduced to support the use of NetStream and Cameras in a manner similar to the use of StageVideo. These textures can be used as source textures in stage3D rendering pipeline. You can use them as rectangular, RGB, or no mipmap textures in rendering of a scene. They are treated as ARGB texture by the shaders which implies that the AGAL shaders do not have to bother about YUV to RGB conversion now. The shaders treat these textures as ARGB textures. This allows you to use the standard shaders with static images without any need for modification. When you render using these textures, the image that is used by the rendering pipeline is the the latest frame at that time. Though, there is no tearing in the video frame, if you use the same texture many times, some of these instances may be picked from different timestamps.

With the use of a VideoTexture object, all this work gets optimized internally - YUV to RGB conversion and texture loading can be completely moved to the GPU. See the VideoTexture devnet article for implementation details.

Note: Video Texture is an existing feature in AIR. It was introduced in AIR 17.0 version.

Windows: Add HiDPI support for AIR desktop (Release)

Beginning with AIR 23, HiDPI support for AIR Desktop on Windows has been introduced. It allows you to provide a higher quality rendering for AIR content on HiDPI displays. This feature works on the concept of higher pixel density scaling instead of pixel-doubling scaling.

Note: HiDPI is already available for AIR on Mac Retina displays.

To implement this feature in an AIR app, the developer should set requestedDisplayResolution as high in the manifest file.

- requestedDisplayResolution set high
  <requestedDisplayResolution>high</requestedDisplayResolution>

Asynchronous Texture Upload for Mobile (Beta)

The feature has been disabled as we iron out few issues and bake it further for release. It will appear back in a future beta.

Till AIR 22, texture upload was synchronous and developers had to wait till the new texture was uploaded successfully.

With AIR 23, you can upload textures asynchronously while the current texture is being rendered. This will ensure better UI performance for AIR applications and smoother user experience.

On successful completion of asynchronous texture upload a TEXTURE_READY event will be generated. Asynchronous upload of texture data for all texture formats for all mip levels is done through two new APIs:

- uploadFromBitmapDataAsync( source:BitmapData, mipmap:uint = 0 )
- uploadFromByteArrayAsync( data:ByteArray, byteArrayOffset:uint, mipmap:uint = 0 )

Mozilla NPAPI AsyncDrawing Support

Async Drawing refers to the method that the browser and Flash Player use to exchange a bitmap surface where Flash Player draws the SWF content. It is used only when the stage is composited with rest of the content in the browser window.

This feature allows wmode “direct” (wmode opaque and transparent) to behave as “windowless” in hardware accelerated async drawing. It is not used in fullscreen mode, or in windowed mode where the plugin draws directly to its own window.

If asynchronous drawing is unavailable for any reason, the plugin falls back to using the existing synchronous drawing model.

AsyncDrawing is supported in NPAPI Plugin on Windows desktop platforms only. It is currently available from FP version 23.0 in Firefox Nightly 51.0a1, the Firefox versions supporting the feature is yet to be announced.

The choice of which Async Drawing path is used (hardware or software) depends on whether the browser supports hardware or software Async Drawing modes.

Following table describes Asynchronous drawing availability by WMODE:
To disable AsynchronousDrawing support in Firefox, go to “about:config” in the search bar of the browser and set “dom.ipc.plugins.asyncdrawing.enabled” to false.

### Runtime Versions

Flash Player: 23.0.0.198
AIR Runtime: 23.0.0.274
AIR SDK & Compiler: 23.0.0.274

### Known Issues

Oct 19, 2016

**Flash Player**
- None

**AIR**
- None

Oct 13, 2016

**Flash Player**
- None

**AIR**
- None

Sept 28, 2016

**Flash Player**
- None

**AIR**
- None

Sept 23, 2016

**Flash Player**
- None

**AIR**
- None

Sept 21, 2016
Flash Player

- [Mac] Flash player quits unexpectedly while playing Forge of Empires on Firefox (4190467).

AIR

- None

Sept 15, 2016

Flash Player

- Resizing embedded flash video player will turn the interface black in xulrunner (4186134)

AIR

- None

Aug 31, 2016

Flash Player

- None

AIR

- None

Aug 24, 2016

Flash Player

- None

AIR

- [iOS] ld-64 Compilation getting failed while packaging app with multiple swf and ANE (4182822)

Aug 10, 2016

Flash Player

- None

AIR

- [Asynchronous Texture Upload] Time lag is observed when a new texture is uploaded asynchronously while the main thread is rendering the texture.

Aug 03, 2016

Flash Player

- [Windows 7] Flash Player quits unexpectedly when SmartErgo app loads the swf (4176988)

AIR

- For HiDPI machines, setting the NativeWindow bound property to a negative value on main monitor results in an increase of NativeWindow bound on the secondary monitor when the main monitor has a greater DPI than the secondary monitor (4176046)

July 27, 2016
Flash Player

- None

AIR

- [Windows] ADL becomes unresponsive when manipulating frame tree of a new frame, before it has been installed in a frame tree (4175625)
- [Android] softKeyboardRect height is incorrect (4085072)
- [iOS 10] Apps are not visible on devices once installed using adt -installApp command
- [iOS] VideoTexture on iOS - sometimes the event TEXTURE_READY isn't dispatched (4150401)
- [iOS] Capabilities.screenDPI gives a wrong value on AIR 22 simulator (4157457)
- [iOS] Context3D.drawToBitmapData uses wrong offset on mobile (4147414)
- [iOS] Crash when trying to access function arguments (4117964)
- [iOS] Landscape mode is not able to initiate the camera and the app crashes in debug mode (4168916)
- [iOS] TextField with TextFieldType.INPUT cuts off text on the left side when focused (4009346)

Fixed Issues

Oct 19, 2016

Flash Player

- None

AIR

- None

Oct 13, 2016

Flash Player

- Flash Player 23.0.0.162 quits unexpectedly on Mac OS 10.11 (4191870)
- Glitches or vertical lines are observed while playing the KingsRoad Game (4189790)
- [Mac]Flash Player quits unexpectedly when it tries to upload the textures through flash.display3D::Context3D/createVertexBuffer(). (4192541)

AIR

- None

Sept 28, 2016

Flash Player

- None

AIR

- Glitches are observed when graphic content is transformed with 2.5D transformation in AIR 23 (4191283)
- Artifacts are observed when Z axis depth of the display object is changed (4189739)
- Triangle is either blank or filled with artifacts when using Graphics.drawTriangles() in AIR 23.0 (4191288)
- Glitches are observed when using 3D rotation is applied on a Bitmap or any containers (4188948)

Sept 23, 2016

Flash Player

- [Mac]Flash player quits unexpectedly while playing Forge of Empires on Firefox (4190467)

AIR

- None
Sept 21, 2016
Flash Player
  - None
AIR
  - None

Sept 15, 2016
Flash Player
  - None
AIR
  - None

Aug 31, 2016
Flash Player
  - None
AIR
  - None

Aug 24, 2016
Flash Player
  - [Windows 7] Adobe Connect Add-In is crashing when Flash player 23 is installed (4182081).
AIR
  - [iOS] StageWebView makes application crash (4180325)
  - [Android] MP4 Video rendering completely broken in Air 22 on all version of Android (4167492)
  - [iOS] Context3D.drawToBitmapData uses wrong offset on mobile (4147414)

Aug 10, 2016
Flash Player
  - None
AIR
  - ATF Textures (with JPEG Compression) are not rendered properly on MALI GPU (3961778)
  - [Windows] TextField contextMenu displaying with offset with HiDPI (a.k.a. Retina) screens (4176561)
  - For HiDPI machines, setting the NativeWindow bound property to a negative value on main monitor results in an increase of NativeWindow bound on the secondary monitor when the main monitor has a greater DPI than the secondary monitor (4176046)

Aug 03, 2016
AIR
  - The "Tab" and "Shift+Tab" keyboard shortcuts do not work on TextField (4174808)
  - [iOS] Capabilities.screenDPI gives a wrong value on AIR 22 simulator (4157457)
  - [iOS] TextField with TextFieldType.INPUT cuts off text on the left when focused (4009346)
  - [Android] Full screen display states must use immersive mode on Android 4.4+ (4100625)

Flash Player
  - None

July 27, 2016
AIR
• [Android] OpenSSL library upgraded to 1.0.2h version
• [iOS] App using concurrency closes when a class is instantiated on the worker .swf (4067035)
• [iOS] CameraRoll on iOS returns Bitmap in incorrect orientation (4070057)
• [iOS] sharedobjects fail when available storage is low (3711301)
• [iOS] [Windows] Error, ld: in section __TEXT,___text reloc 153: R_ABS reloc but no absolute symbol at target address for architecture arm64 while packaging Application with Social.ane using flag -hideAneLibSymbols yes (4005515)

Flash Player

• None

Authoring for Flash Player 23 and AIR 23

• Update application descriptor namespace to 23.
• SWF version should be 34.

System Requirements

For system requirements of the current release of Flash Player in production, visit http://www.adobe.com/products/flashplayer/systemreqs/.
For system requirements of the current release of AIR in production, visit http://www.adobe.com/products/air/systemreqs/.