

AIR CONDITIONING

Lee Brimelow

Platform Evangelist / Adobe





About Me



Lee Brimelow
Platform Evangelist



- ▶ www.gotoandlearn.com
- ▶ www.theflashblog.com



Quick Recap

Adobe[®] AIR[™] lets developers use their **existing** web development skills in HTML, AJAX, Flash and Flex to build and deploy rich Internet applications to the **desktop**.



What You Will Learn

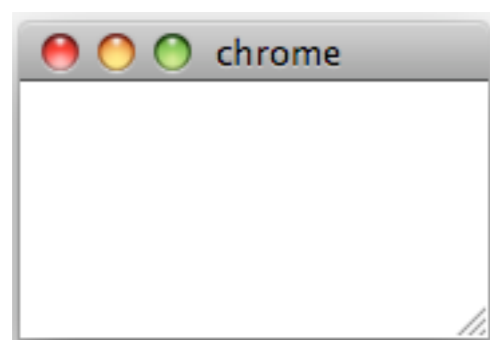
- ▶ Windowing API
- ▶ Menu API
- ▶ Files and Data
- ▶ Drag and Drop
- ▶ Clipboard Interaction
- ▶ SQLite Database
- ▶ OS Integration
- ▶ Network Monitoring



Windowing API



Native Window Types



Normal



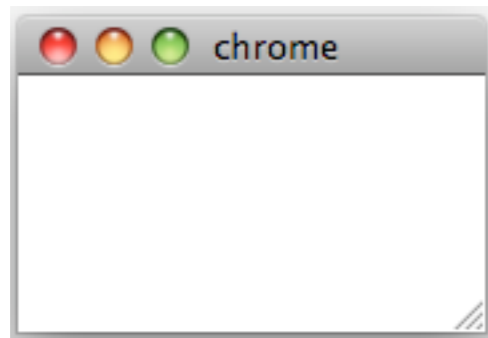
Utility



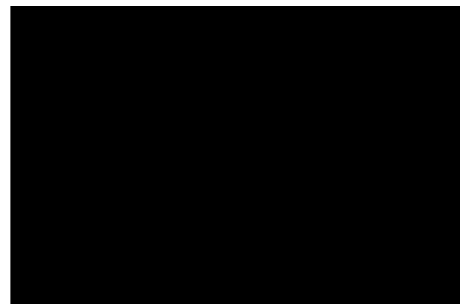
Lightweight



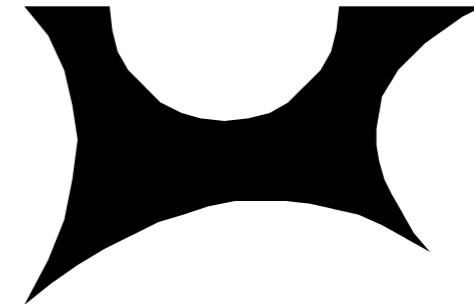
Chrome Options



System



Custom (opaque)



Custom (transparent)



Creating Native Windows

You create native windows using the `NativeWindow` class. The settings for the window are contained in an instance of the `NativeWindowInitOptions` class as shown below:

```
//create the init options
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.transparent = false;
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.type = NativeWindowType.NORMAL;

//create the window
var newWindow:NativeWindow = new NativeWindow(options);
newWindow.title = "My Window";
newWindow.width = 600;
newWindow.height = 400;

//activate and show the new window
newWindow.activate();
```



Manipulating Native Windows

Native window methods:

```
NativeWindow.activate();  
NativeWindow.alwaysInFront = true;  
NativeWindow.close();  
NativeWindow.maximize();  
NativeWindow.minimize();  
NativeWindow.orderInBackOf(otherWin);  
NativeWindow.orderInFrontOf(otherWin);  
NativeWindow.orderToBack();  
NativeWindow.orderToFront();
```

Also has properties like `x`, `y`, `width`, and `height` that can be animated just like a `DisplayObject`.



Adding Content to Windows

You can add content to native windows by simply using the `addChild` method of the native window's `stage` object:

```
//newWindow is a NativeWindow instance
var htmlView:HTMLLoader = new HTMLLoader();
htmlView.width = 300;
htmlView.height = 500;

//set the stage
newWindow.stage.align = "TL";
newWindow.stage.scaleMode = "noScale";
newWindow.stage.addChild(htmlView);

//urlString is the URL of the HTML page to load
htmlView.load(new URLRequest("http://yahoo.com"));
```



Native Window Events

Events in the `flash.events.Event` class:

- ▶ `Event.ACTIVATE`
- ▶ `Event.DEACTIVATE`
- ▶ `Event.CLOSING`
- ▶ `Event.CLOSE`

Events in the `NativeWindowBoundsEvent` class:

- ▶ `NativeWindowBoundsEvent.MOVING`
- ▶ `NativeWindowBoundsEvent.MOVE`
- ▶ `NativeWindowBoundsEvent.RESIZING`
- ▶ `NativeWindowBoundsEvent.RESIZE`

Events in the `NativeWindowDisplayStateEvent` class:

- ▶ `NativeWindowDisplayStateEvent.DISPLAY_STATE_CHANGING`
- ▶ `NativeWindowDisplayStateEvent.DISPLAY_STATE_CHANGE`



Fullscreen Window Mode

You enter full screen mode the same way that you do for a regular SWF by setting the stage's `displayState` property:

```
stage.displayState = StageDisplayState.FULL_SCREEN;
```

Unlike browser SWF files, you can accept keyboard input when in full screen mode. Use responsibly.



Staying On Top

You can make your applications always stay in front of other applications by setting the window's `alwaysInFront` property:

```
NativeWindow.alwaysInFront = true;
```



Handling Multiple Monitors

You can enumerate the screens of the virtual desktop with the following screen methods and properties:

`Screen.screens`

Provides an array of Screen objects describing the available screens. Note that the order of the array is not significant.

`Screen.mainScreen`

Provides a Screen object for the main screen. On Mac OS X, this will be the screen displaying the menu bar. On Windows, this will be the system-designated primary screen.

`Screen.getScreensForRectangle()`

Provides an array of Screen objects describing the screens intersected by the given rectangle. The rectangle passed to this method is in pixel coordinates on the virtual desktop.



Menu API



Native Menu Types

Application menus

An application menu is a global menu that applies to the entire application on OS X.

Window menus

A window menu is associated with a single window and is displayed below the title bar on Windows.

Context menus

Context menus open in response to a right-click or command-click on an interactive object in SWF content or a document element in HTML content.

Dock and system tray icon menus

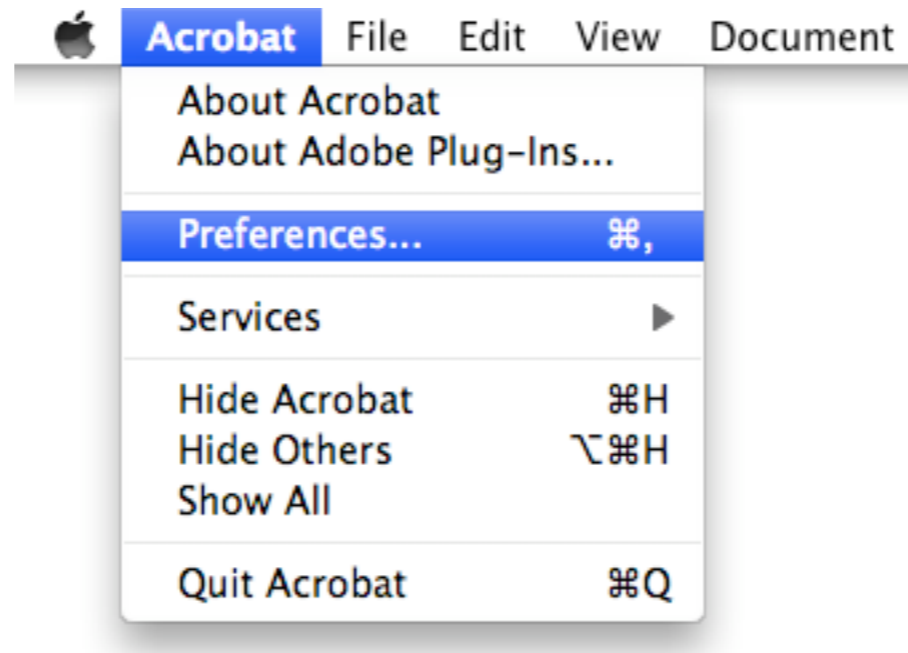
These icon menus are similar to context menus and are assigned to an application icon in the Mac OS X dock or Windows notification area.

Pop-up menus

An AIR pop-up menu is similar to a context menu, but is not necessarily associated with a particular application object or component.



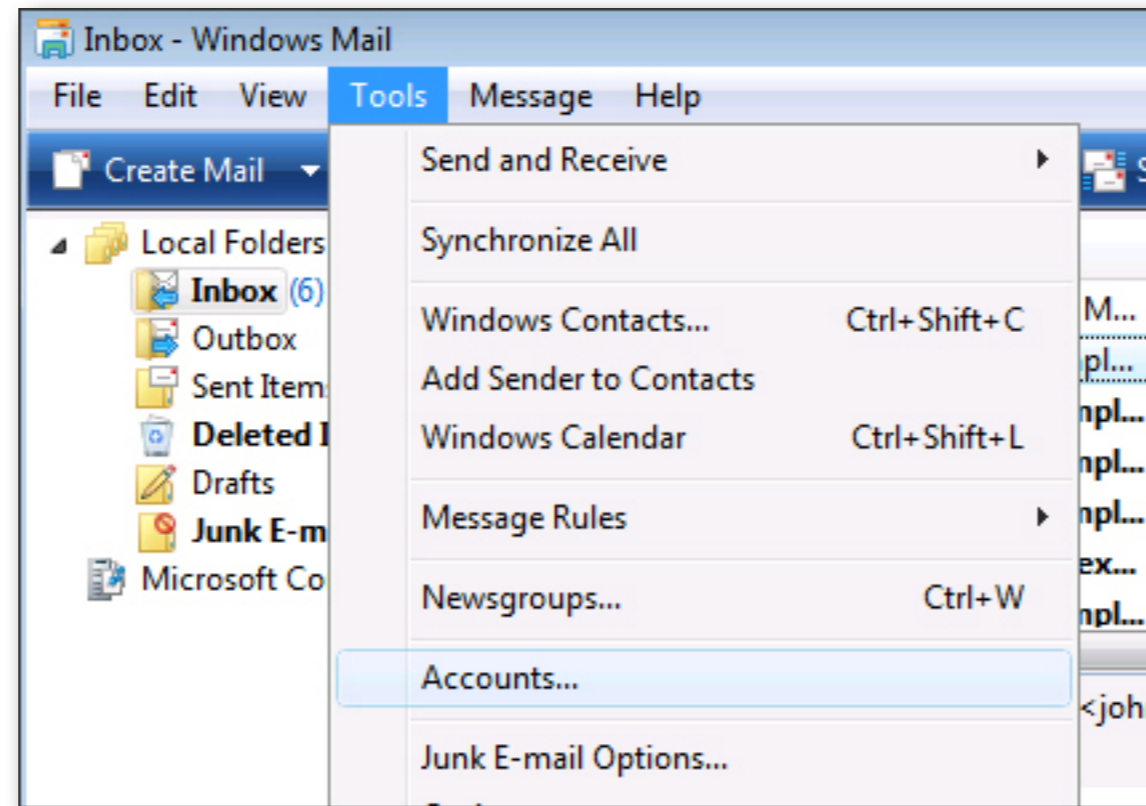
OS X Application Menu



```
var root:NativeMenu = new NativeMenu();  
NativeApplication.nativeApplication.menu = root;
```



Window Menus



```
var root:NativeMenu = new NativeMenu();  
nw.menu = root;
```



Cross-Platform Menus

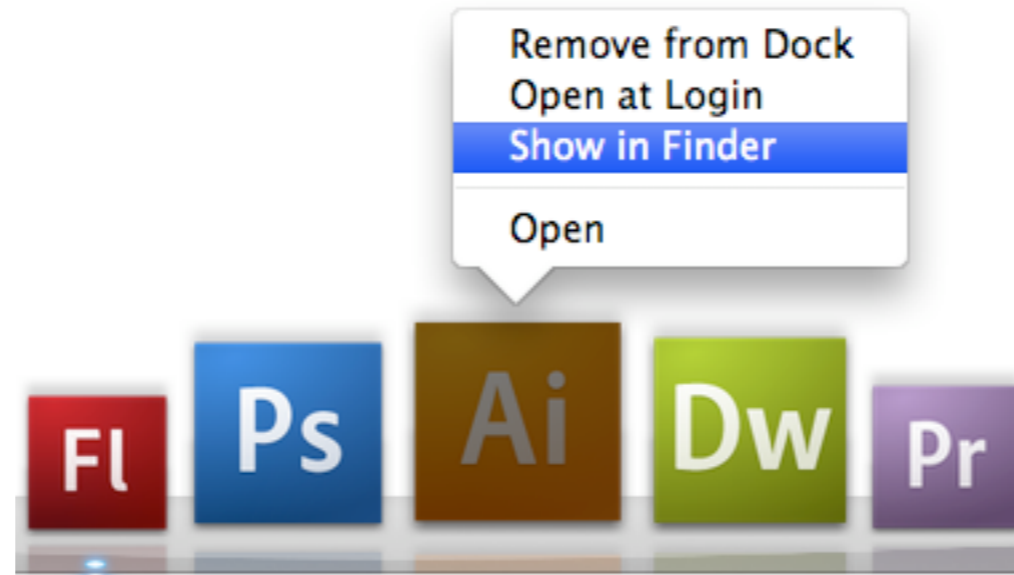


The code shown below can be used to test for OS-specific menu support:

```
if(NativeApplication.supportsMenu)
{
    NativeApplication.nativeApplication.menu.addItem(myMenuItem);
}
else if(NativeWindow.supportsMenu)
{
    var windowMenu:NativeMenu = new NativeMenu();
    this.stage.nativeWindow.menu = windowMenu;
    windowMenu.addItem(myMenuItem);
}
```



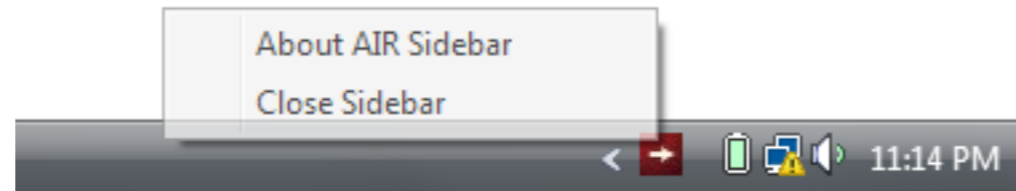
Dock Menus (Mac)



```
var root:NativeMenu = new NativeMenu();  
DockIcon(NativeApplication.nativeApplication.icon).menu = root;
```



System Tray Menus (Win)



```
var root:NativeMenu = new NativeMenu();  
SystemTrayIcon(NativeApplication.nativeApplication.icon).menu = root;
```



Handling Menu Events

Native menus have two important events that you can respond to. The `displaying` event fires right before the menu displays and the `select` event fires when a selection is made as shown below:

```
windowMenu.addEventListener(Event.SELECT, colorChoice);

function colorChoice(event:Event):void
{
    var menuItem:NativeMenuItem = event.target as NativeMenuItem;
    trace(menuItem.label + " has been selected");
}
```



Files and Data



The File Class

- ▶ A `File` object represents a path to a file or directory. This can be an existing file or directory, or it can be one that does not yet exist (for instance, it can represent the path to a file or directory that you wish to create).
- ▶ The `File` class has a number of properties and methods for getting information about the file system and for performing operations (such as copying files and directories).
- ▶ You can use `File` objects along with `FileStream` class to read and write files.
- ▶ The `File` class extends the `FileReference` class. The `FileReference` class, which is available in Flash Player as well as Adobe AIR, represents a pointer to a file, but the `File` class adds properties and methods that are not exposed in Flash Player (in a SWF running in a browser), due to security considerations.



Synchronous vs Asynchronous

Synchronous

The easy way to go. Your application doesn't proceed until the file operation is complete.

Asynchronous

Your application will continue and you will receive events when the file operations are complete. This is a much more flexible approach and is particularly suited for large file operations.



Common File Paths

`File.applicationDirectory`

The folder containing the application's installed files.

`File.applicationStorageDirectory`

The application's private storage directory.

`File.desktopDirectory`

The user's desktop directory.

`File.documentsDirectory`

The user's documents directory.

`File.userDirectory`

The user's directory.



Creating and Deleting Directories

Creating a directory:

```
var dir:File = File.userDirectory.resolvePath("AIR Test");  
dir.createDirectory();
```

Deleting a directory:

```
var dir:File = File.userDirectory.resolvePath("AIR Test");  
dir.deleteDirectory(true);
```



Reading Files

You can read `File` objects using methods of the `FileStream` class as shown in the example below:

```
import flash.filesystem.*;

var file:File = File.documentsDirectory;
file = file.resolvePath("test.txt");
var fileStream:FileStream = new FileStream();
fileStream.open(file, FileMode.READ);
var str:String = fileStream.readMultiByte(file.size, File.systemCharset);
trace(str);
fileStream.close();
```

(The above example uses the synchronous API)



Writing Files

You can write to `File` objects using methods of the `FileStream` class as shown in the example below:

```
import flash.filesystem.*;

var myFile:File = File.documentsDirectory.resolvePath("test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.WRITE);
myFileStream.writeBytes(bytes, 0, bytes.length);
```

(The above example uses the synchronous API)



Encrypted Local Storage

The encrypted local store is valuable for storing settings or authentication information. It uses keychain on the Mac and DPAPI on Windows.

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes);
var storedValue:ByteArray = EncryptedLocalStore.getItem("firstName");
trace(storedValue.readUTFBytes(storedValue.length));
```

The encrypted local data store has an maximum supported total capacity of 10 MB.



Drag and Drop



Drag and Drop Scenarios

Application



Desktop

Application



Application

Desktop



Application



Dragging From the Desktop

You can drag files from the desktop onto an AIR application and easily access the data using the `Clipboard` class.

```
stage.addEventListener(NativeDragEvent.NATIVE_DRAG_ENTER, onDragEnter);
stage.addEventListener(NativeDragEvent.NATIVE_DRAG_DROP, onDragDrop);

function onDragEnter(e:NativeDragEvent):void
{
    NativeDragManager.acceptDragDrop(stage);
}

function onDragDrop(e:NativeDragEvent):void
{
    var files:Object = e.clipboard.getData(ClipboardFormats.FILE_LIST_FORMAT);
}
```



Dragging to the Desktop

You can drag objects from your AIR application onto the desktop using methods of the `Clipboard` class.

```
var bitmapFile:File = new File(fileURL);  
var transferObject:Clipboard = createClipboard(myImage, bitmapFile);  
NativeDragManager.doDrag(this,transferObject,display.bitmapData,  
                           new Point(-mouseX,-mouseY));
```

You can set a custom bitmap to be used as the drag image. Windows always lowers the alpha of these images.



Clipboard Interaction



Reading From the Clipboard

To read the operating system clipboard, call the `getData()` method of the `Clipboard.generalClipboard` object, passing in the name of the format to read:

```
if(Clipboard.generalClipboard.hasFormat(ClipboardFormats.TEXT_FORMAT))  
{  
    var t:String = Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT);  
}
```

The above code also checks to see if the data on the clipboard is text.



Writing to the Clipboard

To write to the clipboard, add the data to the `Clipboard.generalClipboard` object in one or more formats.

```
var textToCopy:String = "Copy to clipboard.";
Clipboard.generalClipboard.clear();
Clipboard.generalClipboard.addData(ClipboardFormats.TEXT_FORMAT, textToCopy, false);
```

In Flash and Flex, you can also use the `System.setClipboard` method.



SQLite Database



Creating a New Database

To create a new database file, you create a `SQLConnection` instance and call its `openAsync()` method. If you pass a `File` instance that refers to a non-existent file location for the reference parameter (the first parameter), the `openAsync()` method will create a new database file at that file location, then open a connection to the newly created database.

```
var conn:SQLConnection = new SQLConnection();
conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);

var dbFile:File = File.applicationStorageDirectory.resolvePath("DBSample.db");
conn.openAsync(dbFile);

function openHandler(event:SQLEvent):void
{
    trace("the database was created successfully");
}
```



Connecting to a Database

```
var conn:SQLConnection = new SQLConnection();
conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, errorHandler);

var dbFile:File = File.applicationStorageDirectory.resolvePath("DBSample.db");
conn.openAsync(dbFile, SQLMode.UPDATE);

function openHandler(event:SQLEvent):void
{
    trace("the database was opened successfully");
}
```

Notice that in the `openAsync()` method call, the second argument is the constant `SQLMode.UPDATE`. Specifying `SQLMode.UPDATE` for the second parameter (`openMode`) causes the runtime to dispatch an error if the specified file doesn't exist.



Running SQL Commands

To retrieve existing data from a database, you create a `SQLStatement` instance, assign the desired SQL `SELECT` statement to the instance's `text` property, then call the `execute()` method.

```
var selectStmt:SQLStatement = new SQLStatement();  
  
// A SQLConnection named "conn" has been created previously  
selectStmt.sqlConnection = conn;  
selectStmt.text = "SELECT itemId, itemName, price FROM products";  
  
selectStmt.addEventListener(SQLEvent.RESULT, resultHandler);  
selectStmt.addEventListener(SQLErrorEvent.ERROR, errorHandler);  
selectStmt.execute();
```



OS Interaction



Launch on Login

An AIR application can be set to launch automatically when the current user logs in by setting the `NativeApplication.nativeApplication.startAtLogin` property to `true`. Once set, the application will automatically start whenever the user logs in until the setting is changed to false, the user manually changes the setting through the operating system, or the application is uninstalled. Launching on login is a run-time setting.

The application does not launch upon system startup. It only starts once the user logs into his/her account.



File Associations

You can set or retrieve filetype information using four methods of the `NativeApplication` class.

`isSetAsDefaultApplication()`

Returns true if the AIR application is currently associated with the specified file type.

`setAsDefaultApplication()`

Creates the association between the AIR application and the open action of the file type.

`removeAsDefaultApplication()`

Removes the association between the AIR application and the file type.

`getDefaultApplication()`

Reports the path of the application that is currently associated with the file type.



Tracking User Presence

```
NativeApplication.nativeApplication.idleThreshold = 120;  
NativeApplication.nativeApplication.addEventListener(Event.USER_IDLE, idleEvent);  
NativeApplication.nativeApplication.addEventListener(Event.USER_PRESENT, presEvent);
```

- ▶ Threshold cannot be shorter than 5 seconds. This is not mentioned in the documentation.
- ▶ In my opinion the results are not as expected. This is an OS-level idle state rather than for this specific application.



Application Termination

The quickest way to terminate an application is to call `NativeApplication.nativeApplication.exit()` and this works fine when your application has no data to save or resources to clean up.

Application termination can be triggered by the user (or OS) in the following ways:

- ▶ Closing the last application window when `NativeApplication.nativeApplication.autoExit` is true.
- ▶ Application exit command from the OS, for example, when the user chooses the exit application command from the default menu.
- ▶ Computer shut down.



Network Monitoring



Network Connectivity

Network Change Event

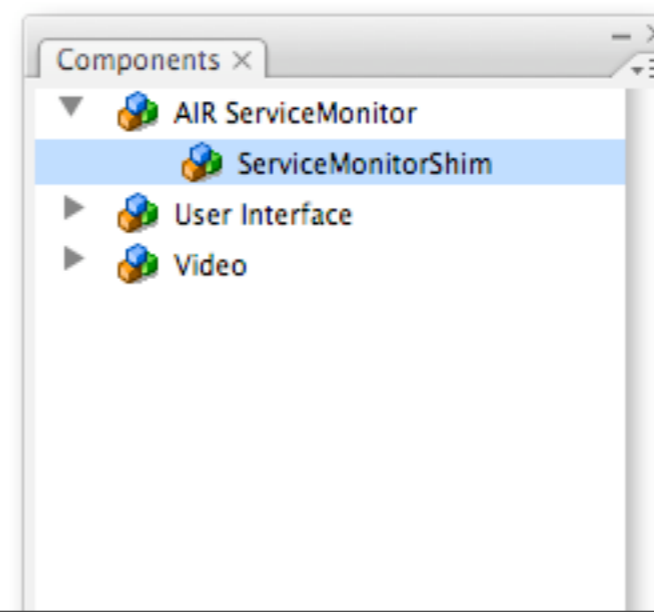
```
NativeApplication.nativeApplication.addEventListener(Event.NETWORK_CHANGE,nc);
```

HTTP Status Detection

```
var monitor:URLMonitor;  
monitor = new URLMonitor(new URLRequest('http://www.adobe.com'));  
monitor.addEventListener(StatusEvent.STATUS, announceStatus);  
monitor.start();
```

(Socket monitor also available)

You need to add the ServiceMonitorShim component to your project in Flash CS3 in order to use the air.net package.





Cross-Application Communication

Use the `LocalConnection` class to communicate between other AIR applications and even with SWF files hosted in the browser.

```
var con:LocalConnection = new LocalConnection();  
con.connect(appId);
```

The `appId` string consists of the string `app#` followed by the application ID as in the example below:

```
con.connect("app#com.leebrimelow.myApp");
```



Learning Resources




gotoAndLearn

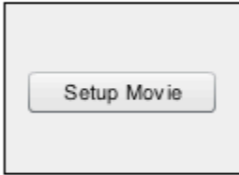
gotoAndLearn ()

FAQS FORUMS THE FLASH BLOG CONTACT

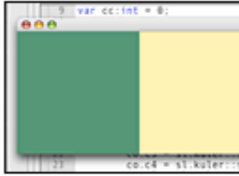
Show: All Tutorials Sort by [Title · Length · Date Added]



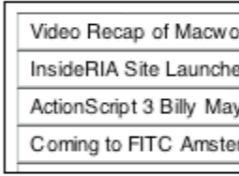
Advanced After Effects and Flash
Learn how to create some slick animated buttons using After Effects and Flash.
Length: 24:52
FL AE Ps PLAY DOWNLOAD FILES




Building Custom Flash Panels
This tutorial shows you how to extend Flash by creating custom panels.
Length: 23:50
FL PLAY DOWNLOAD FILES



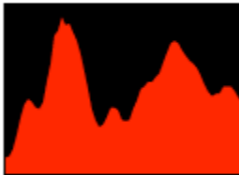
ActionScript 3 Advanced XML
This tutorial shows you advanced XML features while building a Kuler theme viewer.
Length: 24:50
FL Dw PLAY DOWNLOAD FILES



ActionScript 3 XML Basics
This tutorial shows how to use the new XML features in AS3 by creating a simple RSS reader.
Length: 20:09
FL PLAY DOWNLOAD FILES



Advanced Filter Effects
This tutorial shows you how to create some dazzling filter effects with ActionScript 3.
Length: 17:11
FL PLAY DOWNLOAD FILES



Sound Spectrum Display
Learn how to read and display waveform data from your sound files using ActionScript 3.
Length: 20:32
FL PLAY DOWNLOAD FILES

Video tutorials on using Flash and AIR





AIR Homepage

Home / Products /
Adobe AIR

AOL
finetune
eBay
NASDAQ
Pounce

Finetune expanded its popular online music service to the desktop, even including integration with the user's desktop music collection.
[Get application >](#) [Learn more >](#)

Get **ADOBE AIR™**

NEXT STEPS

- Download Adobe AIR now
- Get the free Adobe AIR SDK
- Get AIR applications
- Get support

PART OF THE ADOBE TECHNOLOGY PLATFORM

ADOBE AIR HOME

- Business benefits
- Get started
- Showcase applications
- Browser vs. desktop
- News
- FAQ
- System requirements
- Developer Center
- Download AIR applications

The official AIR site






Christian Cantrell's Blog

Christian Cantrell

Apollo Application Developer



February 25, 2008

What you might not know about AIR (yet)

If you're reading my blog, I'm going to assume that you already know what AIR is. If you don't, have a look at the [official AIR product page](#) before reading any further. Rather than tell you what you probably already know about AIR, I'd like to mention a few things that people might not have realized yet.

AIR isn't just another Adobe product. It isn't just a new tool or utility. It's an entirely new way to develop, deploy, install, and use desktop applications. Entirely new. I really can't stress this point enough. In the world of software, new products are launched all the time (if you don't believe me, just follow [TechCrunch](#) for a few days – it's staggering), each with a marketing team that wants you to believe that their technology is nothing less than revolutionary. So when something completely different does come along, it's sometimes hard to distinguish. How successful AIR will be and how quickly the technology is adopted is an entirely different topic, but there is absolutely no doubt that it's new and disruptive. I could easily write an entire white paper on the importance of AIR, but for the sake of brevity, I'll focus on three main points:

SEARCH

Search this site:

CATEGORIES

- [ActionScript](#)
- [Adobe](#)
- [Apollo](#)
- [ColdFusion](#)
- [Conferences](#)
- [Cool Tools](#)
- [Flash](#)
- [Flex](#)
- [General](#)
- [Java](#)
- [Mac Fanatic](#)

<http://weblogs.macromedia.com/cantrell/>





Adobe AIR Dev Center

Adobe.com Welcome Sign in and join ADC | RSS | United States (Change)

ADOBE DEVELOPER CONNECTION

Search for... All adobe.com Search

PRODUCTS

Acrobat	ColdFusion	Flash	Flex	ActionScript	Architecture	Mobile and devices	Security
Adobe AIR	Dreamweaver	Flash Player	LiveCycle	Ajax	CSS	PDF	Video

TECHNOLOGIES

DEVELOPER RESOURCES

Adobe blogs	Blog aggregator	Documentation	Exchanges	Flex cookbook	Adobe Labs	Support	Trial downloads
Article archive	CSS Advisor	Edge newsletter	Flex bug base	Forums	SDKs	Training	Adobe Open Source

Home / Developer Connection /

Adobe AIR Developer Center


HOME GETTING STARTED SAMPLES DOWNLOADS COMMUNITY

Adobe AIR is a cross-operating system runtime that enables you to use your existing HTML/Ajax, Flex, or Flash web development skills and tools to build and deploy rich Internet applications to the desktop.

Adobe AIR applications support native desktop integration, including clipboard and drag-and-drop support, local file IO, system notification, and more.

[Develop with HTML/Ajax >](#)
[Develop with Flash >](#)
[Develop with Flex >](#)

NICOLAS LIERMAN ON GOOGLE ANALYTICS



Find out what it took to build the [Analytics Reporting Suite](#) for Google Analytics.

NEXT STEPS

- Download Adobe AIR
- Adobe AIR online seminars
- Adobe AIR Marketplace
- Adobe AIR Developer Center RSS feed

ADOBE AIR DEVELOPER CENTER

- Home
- For Flex
- For Flash
- For HTML and Ajax

<http://www.adobe.com/devnet/air/>





On AIR Tour Europe



Coming to Amsterdam on April 4th 2008!

<http://onair.adobe.com>