ADOBE® AIR® 응용 프로그램 만들기

마지막 업데이트 **2017** 년 **5** 월 **31** 일

법적 고지 사항

법적 고지 사항은 http://help.adobe.com/ko_KR/legalnotices/index.html을 참조하십시오.

목차

13장: AIR 응용 프로그램 서명

14장: AIR 응용 프로그램 설명자 파일

15장: 장치 프로파일

16장: AIR.SWF 인 브라우저 API

17장: AIR 응용 프로그램 업데이트

18장: 소스 코드 보기

19장: AIR HTML Introspector를 사용한 디버깅

20장: AIR 응용 프로그램 지역화

21장: path 환경 변수

1장: Adobe AIR

Adobe® AIR®는 멀티 운영 체제에서 실행되는 멀티스크린 런타임으로, 기존 웹 개발 기술을 활용하여 RIA(Rich Internet Application)를 구축하고 데스크톱 및 휴대 장치에 배포할 수 있도록 합니다. Adobe® Flex 및 Adobe® Flash®를 사용하여 ActionScript 3.0으로 데스크톱, TV 및 모바일 AIR 응용 프로그램을 만들 수 있습니다(SWF 기반). 데스크톱 AIR 응용 프로그램은 HTML, JavaScript® 및 Ajax를 사용하여 만들 수도 있습니다(HTML 기반).

Adobe AIR Developer Connection(http://www.adobe.com/devnet/air/)에서 Adobe AIR를 시작하고 사용하는 방법에 대한 자세한 내용을 참조할 수 있습니다.

AIR를 사용하면 친숙한 환경에서 도구 및 응용 프로그램을 가장 편안한 방식으로 활용하면서 작업할 수 있습니다. Flash, Flex, HTML, JavaScript 및 Ajax를 지원하므로 사용자 요구에 가장 적합한 환경을 구축할 수 있습니다.

예를 들어, 다음 기술 중 하나 이상을 사용하여 응용 프로그램을 개발할 수 있습니다.

- Flash / Flex / ActionScript
- HTML / JavaScript / CSS / Ajax

사용자는 기본 응용 프로그램과 상호 작용할 때와 같은 방식으로 AIR 응용 프로그램과 상호 작용합니다. 런타임이 사용자의 컴퓨터 또는 장치에 설치되면 AIR 응용 프로그램이 설치되고 다른 데스크톱 응용 프로그램과 마찬가지로 실행됩니다. iOS에서는 별도의 AIR 런타임이 설치되지 않습니다. 각 iOS AIR 응용 프로그램이 독립 실행형 응용 프로그램입니다.

런타임은 응용 프로그램을 배포하기 위한 일관된 크로스 운영 체제 플랫폼과 프레임워크를 제공하므로 데스크톱 전반에서 일관된 기능과 상호 작용을 보장하여 크로스 브라우저 테스트를 제거합니다. 특정 운영 체제용으로 개발하는 대신 런타임을 대상으로 개발할 수 있으므로 다음과 같은 이점이 있습니다.

- AIR용으로 개발된 응용 프로그램이 추가 작업 없이 여러 운영 체제에서 실행됩니다. 런타임은 AIR에서 지원하는 모든 운영 체제에서 일관되고 예측 가능한 프레젠테이션과 상호 작용을 보장합니다.
- 기존 웹 기술과 디자인 패턴을 활용할 수 있어 응용 프로그램을 빠르게 작성할 수 있습니다. 기존 데스크톱 개발 기술이나 복잡한 기본 코드에 대해 모르더라도 웹 기반 응용 프로그램을 데스크톱으로 확장할 수 있습니다.
- 응용 프로그램 개발이 C 및 C++와 같은 하위 수준 언어를 사용하는 경우보다 쉽습니다. 각 운영 체제와 관련된 복잡한 하위 수준 API를 관리할 필요가 없습니다.

AIR용 응용 프로그램을 개발할 때 프레임워크와 API의 다양한 집합을 활용할 수 있습니다.

- AIR 프레임워크와 런타임에서 제공하는 AIR와 관련된 API
- Flex 프레임워크와 SWF 파일에서 사용되는 ActionScript API 및 다른 ActionScript 기반 라이브러리와 프레임워크
- HTML, CSS 및 JavaScript
- 대부분의 Ajax 프레임워크
- Adobe AIR용 기본 확장은 기본 코드로 프로그래밍된 플랫폼별 기능에 액세스할 수 있도록 하는 ActionScript API를 제공합니다. 또한 기본 확장을 통해 레거시 기본 코드 및 더 높은 성능을 제공하는 기본 코드에 액세스할 수 있습니다.

AIR는 응용 프로그램을 만들고 배포하고 사용하는 방법을 동적으로 변경합니다. 독창적인 제어 기능을 활용하여 Flash, Flex, HTML 및 Ajax 기반 응용 프로그램을 데스크톱, 휴대 장치 및 TV로 확장할 수 있습니다.

각각의 새 AIR 업데이트에 포함된 내용에 대한 자세한 정보는 Adobe AIR 릴리스 정보 (http://www.adobe.com/go/learn_air_relnotes_kr)를 참조하십시오.

2장: Adobe AIR 설치

Adobe® AIR® 런타임을 사용하여 AIR 응용 프로그램을 실행할 수 있습니다. 다음과 같은 방법으로 런타임을 설치할 수 있습니다.

- AIR 응용 프로그램 설치 없이 런타임을 개별적으로 설치합니다.
- 처음으로 웹 페이지 설치 "badge"를 통해 AIR 응용 프로그램을 설치합니다(런타임을 설치하라는 메시지가 표시됨).
- 응용 프로그램과 런타임을 둘 다 설치하는 사용자 정의 설치 프로그램을 만듭니다. 이 방식으로 AIR 런타임을 배포하려면 Adobe로부터 승인을 받아야 합니다. Adobe 런타임 라이센스 페이지에서 승인을 요청할 수 있습니다. Adobe에서는 이러한 설치 프로그램을 작성하기 위한 도구를 제공하지 않습니다. 그러나 다양한 타사 설치 프로그램 도구 키트를 사용할 수 있습니다.
- AIR를 전용 런타임으로 번들하는 AIR 응용 프로그램을 설치합니다. 전용 런타임은 번들 응용 프로그램에만 사용되며, 다른 AIR 응용 프로그램을 실행하는 데는 사용되지 않습니다. 런타임 번들은 Mac 및 Windows에서 옵션입니다. iOS에서는 모든 응용 프로그램에 번들된 런타임이 포함됩니다. AIR 3.7부터는 모든 Android 응용 프로그램에 번들된 런타임이 기본적으로 포함됩니다(별도의 런타임을 사용하는 옵션이 있는 경우에도).
- AIR SDK, Adobe® Flash® Builder™ 또는 Adobe Flex® SDK(AIR 명령줄 개발 도구 포함)와 같은 AIR 개발 환경을 설정합니다. SDK에 포함된 런타임은 응용 프로그램을 디버깅할 때만 사용되며, 설치된 AIR 응용 프로그램을 실행하는 데에는 사용되지 않습니다.

AIR 설치 및 AIR 응용 프로그램 실행을 위한 시스템 요구 사항은 Adobe AIR: 시스템 요구 사항 (http://www.adobe.com/kr/products/air/systemreqs/)을 참조하십시오.

런타임 설치 프로그램 및 AIR 응용 프로그램 설치 프로그램은 모두 AIR 응용 프로그램 또는 AIR 런타임 자체를 설치, 업데이트 또는 제거할 때 로그 파일을 만듭니다. 이러한 로그를 참조하여 설치 문제의 원인을 확인할 수 있습니다. 설치 로그를 참조하십시오.

Adobe AIR 설치

런타임을 설치하거나 업데이트하려면 사용자에게 컴퓨터에 대한 관리 권한이 필요합니다.

Windows 컴퓨터에 런타임 설치

1. http://get.adobe.com/kr/air/에서 런타임 설치 파일을 다운로드합니다.
2. 런타임 설치 파일을 두 번 클릭합니다.
3. 설치 윈도우에서 프롬프트에 따라 설치를 완료합니다.

Mac 컴퓨터에 런타임 설치

1. http://get.adobe.com/kr/air/에서 런타임 설치 파일을 다운로드합니다.
2. 런타임 설치 파일을 두 번 클릭합니다.
3. 설치 윈도우에서 프롬프트에 따라 설치를 완료합니다.
4. 설치 프로그램에서 인증 윈도우를 표시하면 해당 Mac OS 사용자 이름과 암호를 입력합니다.

Linux 컴퓨터에서 런타임 설치

참고: 현재 AIR 2.7 이상은 Linux에서 지원되지 않습니다. Linux에 배포된 AIR 응용 프로그램은 계속 AIR 2.6 SDK를 사용해야 합니다.

이진 설치 프로그램 사용:

1 http://kb2.adobe.com/kr/cps/853/cpsid_85304.html에서 설치 이진 파일을 찾아 다운로드합니다.

2 설치 프로그램을 실행할 수 있도록 파일 권한을 설정합니다. 명령줄에서 다음을 사용하여 파일 사용 권한을 설정할 수 있습니다.

```
chmod +x AdobeAIRInstaller.bin
```

일부 버전의 Linux에서는 컨텍스트 메뉴를 통해 여는 [속성] 대화 상자에서 파일 권한을 설정할 수 있습니다.

3 런타임 설치 파일을 두 번 클릭하거나 명령줄에서 설치 프로그램을 실행합니다.

4 설치 윈도우에서 프롬프트에 따라 설치를 완료합니다.

Adobe AIR가 기본 패키지로 설치됩니다. 즉, RPM 기반 배포에서는 rpm으로, Debian 배포에서는 deb로 설치됩니다. 현재 AIR에서 다른 패키지 형식은 지원하지 않습니다.

패키지 설치 프로그램 사용:

1 http://kb2.adobe.com/kr/cps/853/cpsid_85304.html에서 AIR 패키지 파일을 찾습니다. 시스템에서 지원하는 패키지 형식에 따라 RPM 또는 Debian 패키지를 다운로드합니다.

2 필요한 경우 AIR 패키지 파일을 두 번 클릭하여 패키지를 설치합니다.

또한 명령줄에서 다음과 같이 설치할 수 있습니다.

a Debian 시스템:

```
sudo dpkg -i <path to the package>/adobeair-2.0.0.xxxxx.deb
```

b RPM 기반 시스템:

```
sudo rpm -i <path to the package>/adobeair-2.0.0-xxxxx.i386.rpm
```

또는 기존 버전(AIR 1.5.3 이상)을 업데이트하는 경우:

```
sudo rpm -U <path to the package>/adobeair-2.0.0-xxxxx.i386.rpm
```

AIR 2 및 AIR 응용 프로그램을 설치하려면 컴퓨터에 대한 관리자 권한이 있어야 합니다.

Adobe AIR는 /opt/Adobe AIR/Versions/1.0에 설치됩니다.

AIR는 MIME 형식 "application/vnd.adobe.air-application-installer-package+zip"을 등록하는데, 이는 .air 파일이 MIME 형식이며 AIR 런타임에 등록된다는 것을 나타냅니다.

Android 장치에 런타임 설치

Android Market으로부터 AIR 런타임의 최신 릴리스를 설치할 수 있습니다.

웹 페이지에 있는 링크로부터 또는 ADT -installRuntime 명령을 사용하여 AIR 런타임의 개발용 버전을 설치할 수 있습니다. 한 번에 한 버전의 AIR 런타임만 설치할 수 있습니다. 릴리스와 개발용 버전을 둘 다 설치할 수는 없습니다.

자세한 내용은 159페이지의 “ADT installRuntime 명령”을 참조하십시오.

iOS 장치에 런타임 설치

필수 AIR 런타임 코드는 iPhone, iTouch 및 iPad 장치에 대해 생성되는 각 응용 프로그램에 번들로 포함됩니다. 별도의 런타임 구성 요소는 설치하지 않습니다.

기타 도움말 항목

63페이지의 “AIR for iOS”

Adobe AIR 제거

런타임을 설치하면 다음 절차에 따라 제거할 수 있습니다.

Windows 컴퓨터에서 런타임 제거

1. Windows [시작] 메뉴에서 [설정] > [제어판]을 선택합니다.
2. [프로그램], [프로그램 또는 기능] 또는 [프로그램 추가 또는 제거] 제어판을 엽니다(실행 중인 Windows의 버전에 따라 다름).
3. "Adobe AIR"를 선택하여 런타임을 제거합니다.
4. [변경/제거] 버튼을 클릭합니다.

Mac 컴퓨터에서 런타임 제거

- 응용 프로그램/유틸리티 폴더에서 "Adobe AIR Uninstaller"를 두 번 클릭합니다.

Linux 컴퓨터에서 런타임 제거

다음 중 하나를 수행합니다.

- [응용 프로그램] 메뉴에서 "Adobe AIR Uninstaller" 명령을 선택합니다.
- -uninstall 옵션을 사용하여 AIR 설치 프로그램 이진을 실행합니다.
- 패키지 관리자를 사용하여 AIR 패키지(adobeair 및 adobecerts)를 제거합니다.

Android 장치에서 런타임 제거

1. 장치에서 설정 응용 프로그램을 엽니다.
2. [응용 프로그램] > [응용 프로그램 관리] 아래에서 Adobe AIR 항목을 누릅니다.
3. [제거] 버튼을 누릅니다.

ADT -uninstallRuntime 명령을 사용할 수도 있습니다. 자세한 내용은 160페이지의 “ADT uninstallRuntime 명령”을 참조하십시오.

번들된 런타임 제거

번들된 전용 런타임을 제거하려면 함께 설치된 응용 프로그램을 제거해야 합니다. 전용 런타임은 해당 설치 응용 프로그램을 실행하는 데만 사용됩니다.

AIR 샘플 응용 프로그램 설치 및 실행

AIR 응용 프로그램을 설치하거나 업데이트하려면 컴퓨터에 대한 관리자 권한이 있어야 합니다.

AIR 기능을 보여 주는 일부 샘플 응용 프로그램을 사용할 수 있습니다. 다음 지침에 따라 이러한 응용 프로그램에 액세스하고 설치할 수 있습니다.

1. AIR 샘플 응용 프로그램을 다운로드하고 실행합니다. 컴파일된 응용 프로그램과 소스 코드를 사용할 수 있습니다.
2. 샘플 응용 프로그램을 다운로드하고 실행하려면 샘플 응용 프로그램의 "Install Now" 버튼을 클릭합니다. 응용 프로그램을 설치하고 실행하라는 프롬프트가 표시됩니다.

3 샘플 응용 프로그램을 다운로드하고 나중에 실행하도록 선택하는 경우 다운로드 링크를 선택합니다. 다음 방법을 사용하여 언제든지 AIR 응용 프로그램을 실행할 수 있습니다.

- Windows에서는 데스크톱에서 응용 프로그램 아이콘을 두 번 클릭하거나 Windows [시작] 메뉴에서 해당 응용 프로그램을 선택합니다.
- Mac OS에서는 기본적으로 사용자 디렉토리의 Applications 폴더(예: Macintosh HD/Users/JoeUser/Applications/)에 설치되어 있는 응용 프로그램 아이콘을 두 번 클릭합니다.

참고: 이러한 지침의 업데이트는 AIR 릴리스 정보(위치: http://www.adobe.com/go/learn_air_relnotes_kr)를 확인하십시오.

Adobe AIR 업데이트

Adobe는 새로운 기능 또는 문제 해결을 포함하도록 Adobe AIR를 정기적으로 업데이트합니다. 자동 알림 및 업데이트 기능을 사용하면 Adobe AIR의 업데이트된 버전이 제공될 때 Adobe에서 사용자에게 자동으로 알려 줍니다.

Adobe AIR 업데이트는 Adobe AIR가 올바르게 작동하도록 하며 중요한 보안 변경 사항을 포함한 경우가 많습니다. 사용자는 새로운 버전이 제공될 때마다 최신 버전의 Adobe AIR로 업데이트하는 것이 좋습니다. 특히 보안 업데이트가 포함된 경우에는 반드시 업데이트해야 합니다.

기본적으로 AIR 응용 프로그램이 실행되면 런타임에서 업데이트를 사용할 수 있는지 확인합니다. 런타임에서는 마지막 업데이트 확인 후 2주 이상 경과한 경우 업데이트가 있는지 확인합니다. 업데이트를 사용할 수 있는 경우 AIR는 업데이트를 백그라운드에서 다운로드합니다.

사용자는 AIR SettingsManager 응용 프로그램을 사용하여 자동 업데이트 기능을 비활성화할 수 있습니다. AIR SettingsManager 응용 프로그램은 http://airdownload.adobe.com/air/applications/SettingsManager/SettingsManager.air에서 다운로드할 수 있습니다.

일반적으로 Adobe AIR 설치 프로세스 동안 http://airinstall.adobe.com에 연결하면 운영 체제 버전, 언어 등의 설치 환경에 대한 기본적인 정보가 전송됩니다. 이 정보는 설치할 때마다 한 번만 전송되며 Adobe는 이를 통해 설치가 성공적으로 수행되었는지 확인할 수 있습니다. 이때 어떠한 개인 정보도 수집 또는 전송되지 않습니다.

전용 런타임 업데이트

응용 프로그램을 전용 런타임 번들과 함께 배포한 경우 전용 런타임은 자동으로 업데이트되지 않습니다. 사용자의 보안을 위해 Adobe가 게시한 업데이트를 모니터링하고 관련 보안 변경이 게시된 경우 새 런타임 버전으로 응용 프로그램을 업데이트해야 합니다.

3장: AIR API를 사용한 작업

Adobe® AIR®의 일부 기능은 Adobe® Flash® Player에서 실행되는 SWF 내용에 사용할 수 없습니다.

ActionScript 3.0 개발자
Adobe AIR API는 다음 두 문서에 설명되어 있습니다.

- ActionScript 3.0 개발자 안내서
- Adobe Flash Platform용 ActionScript 3.0 참조 설명서

HTML 개발자
HTML 기반 AIR 응용 프로그램을 만드는 경우 AIRAliases.js 파일을 통해 JavaScript에서 사용할 수 있는 API(JavaScript에서 AIR API 클래스 액세스 참조)가 다음 두 문서에 설명되어 있습니다.

- HTML Developer's Guide for Adobe AIR
- Adobe AIR API Reference for HTML Developers

AIR에 고유한 ActionScript 3.0 클래스

다음 표에는 Adobe AIR에 고유한 런타임 클래스가 포함되어 있습니다. 브라우저의 Adobe® Flash® Player에서 실행되는 SWF 내용에는 해당 클래스를 사용할 수 없습니다.

HTML 개발자
AIRAliases.js 파일을 통해 JavaScript에서 사용할 수 있는 클래스는 Adobe AIR API Reference for HTML Developers에 설명되어 있습니다.

클래스	ActionScript 3.0 패키지	AIR 버전에 추가됨
ARecord	flash.net.dns	2.0
AAAARecord	flash.net.dns	2.0
ApplicationUpdater	air.update	1.5
ApplicationUpdaterUI	air.update	1.5
AudioPlaybackMode	flash.media	3.0
AutoCapitalize	flash.text	3.0
BrowserInvokeEvent	flash.events	1.0
CameraPosition	flash.media	3.0
CameraRoll	flash.media	2.0
CameraRollBrowseOptions	flash.media	3.0
CameraUI	flash.media	2.5
CertificateStatus	flash.security	2.0

클래스	ActionScript 3.0 패키지	AIR 버전에 추가됨
CompressionAlgorithm	flash.utils	1.0
DatagramSocket	flash.net	2.0
DatagramSocketDataEvent	flash.events	2.0
DNSResolver	flash.net.dns	2.0
DNSResolverEvent	flash.events	2.0
DockIcon	flash.desktop	1.0
DownloadErrorEvent	air.update.events	1.5
DRMAuthenticateEvent	flash.events	1.0
DRMDeviceGroup	flash.net.drm	3.0
DRMDeviceGroupErrorEvent	flash.net.drm	3.0
DRMDeviceGroupEvent	flash.net.drm	3.0
DRMManagerError	flash.errors	1.5
EncryptedLocalStore	flash.data	1.0
ExtensionContext	flash.external	2.5
File	flash.filesystem	1.0
FileListEvent	flash.events	1.0
FileMode	flash.filesystem	1.0
FileStream	flash.filesystem	1.0
FocusDirection	flash.display	1.0
GameInput	flash.ui	3.0
GameInputControl	flash.ui	3.0
GameInputControlType	flash.ui	3.6 이하. 3.7부터는 삭제됨
GameInputDevice	flash.ui	3.0
GameInputEvent	flash.ui	3.0
GameInputFinger	flash.ui	3.6 이하. 3.7부터는 삭제됨
GameInputHand	flash.ui	3.6 이하. 3.7부터는 삭제됨
Geolocation	flash.sensors	2.0
GeolocationEvent	flash.events	2.0
HTMLHistoryItem	flash.html	1.0
HTMLHost	flash.html	1.0
HTMLLoader	flash.html	1.0
HTMLPDFCapability	flash.html	1.0
HTMLSWFCapabiltiy	flash.html	2.0
HTMLUncaughtScriptExceptionEvent	flash.events	1.0

클래스	ActionScript 3.0 패키지	AIR 버전에 추가됨
HTMLWindowCreateOptions	flash.html	1.0
Icon	flash.desktop	1.0
IFilePromise	flash.desktop	2.0
ImageDecodingPolicy	flash.system	2.6
InteractiveIcon	flash.desktop	1.0
InterfaceAddress	flash.net	2.0
InvokeEvent	flash.events	1.0
InvokeEventReason	flash.desktop	1.5.1
IPVersion	flash.net	2.0
IURIDereferencer	flash.security	1.0
LocationChangeEvent	flash.events	2.5
MediaEvent	flash.events	2.5
MediaPromise	flash.media	2.5
MediaType	flash.media	2.5
MXRecord	flash.net.dns	2.0
NativeApplication	flash.desktop	1.0
NativeDragActions	flash.desktop	1.0
NativeDragEvent	flash.events	1.0
NativeDragManager	flash.desktop	1.0
NativeDragOptions	flash.desktop	1.0
NativeMenu	flash.display	1.0
NativeMenuItem	flash.display	1.0
NativeProcess	flash.desktop	2.0
NativeProcessExitEvent	flash.events	2.0
NativeProcessStartupInfo	flash.desktop	2.0
NativeWindow	flash.display	1.0
NativeWindowBoundsEvent	flash.events	1.0
NativeWindowDisplayState	flash.display	1.0
NativeWindowDisplayStateEvent	flash.events	1.0
NativeWindowInitOptions	flash.display	1.0
NativeWindowRenderMode	flash.display	3.0
NativeWindowResize	flash.display	1.0
NativeWindowSystemChrome	flash.display	1.0
NativeWindowType	flash.display	1.0

클래스	ActionScript 3.0 패키지	AIR 버전에 추가됨
NetworkInfo	flash.net	2.0
NetworkInterface	flash.net	2.0
NotificationType	flash.desktop	1.0
OutputProgressEvent	flash.events	1.0
PaperSize	flash.printing	2.0
PrintMethod	flash.printing	2.0
PrintUIOptions	flash.printing	2.0
PTRRecord	flash.net.dns	2.0
ReferencesValidationSetting	flash.security	1.0
ResourceRecord	flash.net.dns	2.0
RevocationCheckSettings	flash.security	1.0
Screen	flash.display	1.0
ScreenMouseEvent	flash.events	1.0
SecureSocket	flash.net	2.0
SecureSocketMonitor	air.net	2.0
ServerSocket	flash.net	2.0
ServerSocketConnectEvent	flash.events	2.0
ServiceMonitor	air.net	1.0
SignatureStatus	flash.security	1.0
SignerTrustSettings	flash.security	1.0
SocketMonitor	air.net	1.0
SoftKeyboardType	flash.text	3.0
SQLCollationType	flash.data	1.0
SQLColumnNameStyle	flash.data	1.0
SQLColumnSchema	flash.data	1.0
SQLConnection	flash.data	1.0
SQLError	flash.errors	1.0
SQLErrorEvent	flash.events	1.0
SQLErrorOperation	flash.errors	1.0
SQLEvent	flash.events	1.0
SQLIndexSchema	flash.data	1.0
SQLMode	flash.data	1.0
SQLResult	flash.data	1.0
SQLSchema	flash.data	1.0

클래스	ActionScript 3.0 패키지	AIR 버전에 추가됨
SQLSchemaResult	flash.data	1.0
SQLStatement	flash.data	1.0
SQLTableSchema	flash.data	1.0
SQLTransactionLockType	flash.data	1.0
SQLTriggerSchema	flash.data	1.0
SQLUpdateEvent	flash.events	1.0
SQLViewSchema	flash.data	1.0
SRVRecord	flash.net.dns	2.0
StageAspectRatio	flash.display	2.0
StageOrientation	flash.display	2.0
StageOrientationEvent	flash.events	2.0
StageText	flash.text	3.0
StageTextInitOptions	flash.text	3.0
StageWebView	flash.media	2.5
StatusFileUpdateErrorEvent	air.update.events	1.5
StatusFileUpdateEvent	air.update.events	1.5
StatusUpdateErrorEvent	air.update.events	1.5
StatusUpdateEvent	air.update.events	1.5
StorageVolume	flash.filesystem	2.0
StorageVolumeChangeEvent	flash.events	2.0
StorageVolumeInfo	flash.filesystem	2.0
SystemIdleMode	flash.desktop	2.0
SystemTrayIcon	flash.desktop	1.0
TouchEventIntent	flash.events	3.0
UpdateEvent	air.update.events	1.5
Updater	flash.desktop	1.0
URLFilePromise	air.desktop	2.0
URLMonitor	air.net	1.0
URLRequestDefaults	flash.net	1.0
XMLSignatureValidator	flash.security	1.0

AIR 고유 기능이 포함된 Flash Player 클래스

브라우저에서 실행되는 SWF 내용에서 다음과 같은 클래스를 사용할 수 있지만 AIR는 추가 속성 또는 메서드를 제공합니다.

패키지	클래스	속성, 메서드 또는 이벤트	AIR 버전에 추가됨
flash.desktop	Clipboard	supportsFilePromise	2.0
	ClipboardFormats	BITMAP_FORMAT	1.0
		FILE_LIST_FORMAT	1.0
		FILE_PROMISE_LIST_FORMAT	2.0
		URL_FORMAT	1.0
flash.display	LoaderInfo	childSandboxBridge	1.0
		parentSandboxBridge	1.0
	Stage	assignFocus()	1.0
		autoOrients	2.0
		deviceOrientation	2.0
		nativeWindow	1.0
		orientation	2.0
		orientationChange 이벤트	2.0
		orientationChanging 이벤트	2.0
		setAspectRatio	2.0
		setOrientation	2.0
		softKeyboardRect	2.6
		supportedOrientations	2.6
		supportsOrientationChange	2.0
	NativeWindow	owner	2.6
		listOwnedWindows	2.6
	NativeWindowInitOptions	owner	2.6

패키지	클래스	속성, 메서드 또는 이벤트	AIR 버전에 추가됨
flash.events	Event	CLOSING	1.0
		DISPLAYING	1.0
		PREPARING	2.6
		EXITING	1.0
		HTML_BOUNDS_CHANGE	1.0
		HTML_DOM_INITIALIZE	1.0
		HTML_RENDER	1.0
		LOCATION_CHANGE	1.0
		NETWORK_CHANGE	1.0
		STANDARD_ERROR_CLOSE	2.0
		STANDARD_INPUT_CLOSE	2.0
		STANDARD_OUTPUT_CLOSE	2.0
		USER_IDLE	1.0
		USER_PRESENT	1.0
	HTTPStatusEvent	HTTP_RESPONSE_STATUS	1.0
		responseHeaders	1.0
		responseURL	1.0
	KeyboardEvent	commandKey	1.0
		controlKey	1.0

패키지	클래스	속성, 메서드 또는 이벤트	AIR 버전에 추가됨
flash.net	FileReference	extension	1.0
		httpResponseStatus 이벤트	1.0
		uploadUnencoded()	1.0
	NetStream	drmAuthenticate 이벤트	1.0
		onDRMContentData 이벤트	1.5
		preloadEmbeddedData()	1.5
		resetDRMVouchers()	1.0
		setDRMAuthenticationCredentials()	1.0
	URLRequest	authenticate	1.0
		cacheResponse	1.0
		followRedirects	1.0
		idleTimeout	2.0
		manageCookies	1.0
		useCache	1.0
		userAgent	1.0
	URLStream	httpResponseStatus 이벤트	1.0
flash.printing	PrintJob	active	2.0
		copies	2.0
		firstPage	2.0
		isColor	2.0
		jobName	2.0
		lastPage	2.0
		maxPixelsPerInch	2.0
		paperArea	2.0
		printableArea	2.0
		printer	2.0
		printers	2.0
		selectPaperSize()	2.0
		showPageSetupDialog()	2.0
		start2()	2.0
		supportsPageSetupDialog	2.0
		terminate()	2.0
	PrintJobOptions	pixelsPerInch	2.0
		printMethod	2.0

패키지	클래스	속성, 메서드 또는 이벤트	AIR 버전에 추가됨
flash.system	Capabilities	languages	1.1
	LoaderContext	allowLoadBytesCodeExecution	1.0
	Security	APPLICATION	1.0
flash.ui	KeyLocation	D_PAD	2.5

이러한 새 속성 및 메서드의 대부분은 AIR 응용 프로그램 보안 샌드박스의 내용에서만 사용할 수 있습니다. 그러나 기타 샌드박스에서 실행되는 내용에서는 URLRequest 클래스의 새 멤버도 사용할 수 있습니다.

ByteArray.compress() 및 ByteArray.uncompress() 메서드에는 각각 algorithm 매개 변수가 포함되어 있으므로 deflate 압축과 zlib 압축 중에서 선택할 수 있습니다. 이 매개 변수는 AIR에서 실행되는 내용에만 사용할 수 있습니다.

AIR 고유 Flex 구성 요소

Adobe AIR에 대한 내용을 개발할 때 다음과 같은 Adobe® Flex™ MX 구성 요소를 사용할 수 있습니다.

- FileEvent
- FileSystemComboBox
- FileSystemDataGrid
- FileSystemEnumerationMode
- FileSystemHistoryButton
- FileSystemList
- FileSystemSizeDisplayMode
- FileSystemTree
- FlexNativeMenu
- HTML
- Window
- WindowedApplication
- WindowedSystemManager

또한 Flex 4에는 다음과 같은 Spark AIR 구성 요소가 포함되어 있습니다.

- Window
- WindowedApplication

AIR Flex 구성 요소에 대한 자세한 내용은 Using the Flex AIR components를 참조하십시오.

4장: AIR 개발용 Adobe Flash Platform 도구

다음과 같은 Adobe Flash Platform 개발 도구를 사용하여 AIR 응용 프로그램을 개발할 수 있습니다.

ActionScript 3.0(Flash 및 Flex) 개발자용:

- Adobe Flash Professional(AIR용으로 제작 참조)
- Adobe Flex 3.x 및 4.x SDK(17페이지의 "Flex SDK 설정" 및 149페이지의 "ADT(AIR Developer Tool)" 참조)
- Adobe Flash Builder(Flash Builder를 사용하여 AIR 응용 프로그램 개발 참조)

HTML 및 Ajax 개발자용:

- Adobe AIR SDK (15페이지의 "AIR SDK 설치" 및 149페이지의 "ADT(AIR Developer Tool)" 참조)
- Adobe Dreamweaver CS3, CS4, CS5(AIR Extension for Dreamweaver 참조)

AIR SDK 설치

Adobe AIR SDK에는 응용 프로그램 실행 및 패키지를 위해 사용하는 다음과 같은 명령줄 도구가 포함됩니다.

ADL(AIR Debug Launcher) AIR 응용 프로그램을 먼저 설치하지 않고도 실행할 수 있게 해줍니다. 144페이지의 "ADL(AIR Debug Launcher)"을 참조하십시오.

ADT(AIR Development Tool) AIR 응용 프로그램을 배포 가능한 설치 패키지로 패키지화합니다. 149페이지의 "ADT(AIR Developer Tool)"를 참조하십시오.

AIR 명령줄 도구를 사용하려면 Java가 컴퓨터에 설치되어 있어야 합니다. JRE 또는 JDK(버전 1.5 이상)에서 Java 가상 시스템을 사용할 수 있습니다. Java JRE 및 Java JDK는 http://java.sun.com/에서 제공됩니다.

ADT 도구를 실행하려면 최소한 2GB의 컴퓨터 메모리가 필요합니다.

참고: 최종 사용자가 AIR 응용 프로그램을 실행할 때는 Java가 필요하지 않습니다.

AIR SDK로 AIR 응용 프로그램을 만드는 방법에 대한 간단한 개요를 보려면 30페이지의 "AIR SDK를 사용하여 첫 번째 HTML 기반 AIR 응용 프로그램 만들기"를 참조하십시오.

AIR SDK 다운로드 및 설치

다음 지침에 따라 AIR SDK를 다운로드하고 설치할 수 있습니다.

Windows에 AIR SDK 설치

- AIR SDK 설치 파일을 다운로드합니다.
- AIR SDK는 표준 파일 아카이브로 배포됩니다. AIR를 설치하려면 SDK의 내용을 컴퓨터 폴더에 추출합니다(예: C:\Program Files\Adobe\AIRSDK 또는 C:\AIRSDK).
- ADL 및 ADT 도구는 AIR SDK의 bin 폴더에 포함됩니다. 이 폴더에 대한 경로를 PATH 환경 변수에 추가하십시오.

Mac OS X에 AIR SDK 설치

- AIR SDK 설치 파일을 다운로드합니다.
- AIR SDK는 표준 파일 아카이브로 배포됩니다. AIR를 설치하려면 SDK의 내용을 컴퓨터 폴더에 추출합니다(예: /Users/<사용자 이름>/Applications/AIRSDK).

- ADL 및 ADT 도구는 AIR SDK의 bin 폴더에 포함됩니다. 이 폴더에 대한 경로를 PATH 환경 변수에 추가하십시오.

Linux에 AIR SDK 설치

- SDK는 tbz2 형식으로 제공됩니다.
- SDK를 설치하려면 SDK의 압축을 해제하려는 폴더를 만든 후 tar -jxvf <path to AIR-SDK.tbz2> 명령을 사용합니다.

AIR SDK 도구 사용 시작에 대한 자세한 내용은 명령줄 도구를 사용하여 AIR 응용 프로그램 만들기를 참조하십시오.

AIR SDK에 포함된 내용

다음 표에서는 AIR SDK에 포함된 파일의 목적에 대해 설명합니다.

SDK 폴더	파일/도구 설명
bin	ADL(AIR Debug Launcher)을 사용하면 AIR 응용 프로그램을 먼저 패키지화하거나 설치하지 않고도 실행할 수 있습니다. 이 도구를 사용하는 방법에 대한 자세한 내용은 144페이지의 "ADL(AIR Debug Launcher)"을 참조하십시오. ADT(AIR Developer Tool)는 응용 프로그램을 배포할 수 있도록 AIR 파일로 패키지화합니다. 이 도구를 사용하는 방법에 대한 자세한 내용은 149페이지의 "ADT(AIR Developer Tool)"를 참조하십시오.
frameworks	libs 디렉토리에는 AIR 응용 프로그램에 사용할 코드 라이브러리가 포함되어 있습니다. projects 디렉토리에는 컴파일된 SWF 및 SWC 라이브러리에 대한 코드가 포함되어 있습니다.
include	include 디렉토리에는 기본 확장을 작성하기 위한 C 언어 헤더 파일이 포함되어 있습니다.
install	install 디렉토리에는 Android 장치용 Windows USB 드라이버가 포함되어 있습니다. Android SDK에서 Google이 제공하는 드라이버입니다.
lib	AIR SDK 도구의 지원 코드가 포함되어 있습니다.
runtimes	데스크톱 및 휴대 장치를 위한 AIR 런타임입니다. 데스크톱 런타임은 ADL이 AIR 응용 프로그램을 패키지화 또는 설치하기 전에 실행하기 위해 사용됩니다. Android용 AIR 런타임(APK 패키지)은 개발 및 테스트를 위해 Android 장치나 에뮬레이터에 설치할 수 있습니다. 장치 및 에뮬레이터에 대해 별도의 APK 패키지가 사용됩니다. Android용 공용 AIR 런타임은 Android Market에서 얻을 수 있습니다.
samples	이 폴더에는 샘플 응용 프로그램 설명자 파일, 간편한 설치 기능 샘플(badge.swf) 및 기본 AIR 응용 프로그램 아이콘이 포함됩니다.
templates	descriptor-template.xml - 각 AIR 응용 프로그램에 필요한 응용 프로그램 설명자 파일의 템플릿입니다. 응용 프로그램 설명자 파일에 대한 자세한 설명은 183페이지의 "AIR 응용 프로그램 설명자 파일"을 참조하십시오. 각 릴리스 버전의 AIR를 위한 응용 프로그램 설명자의 XML 구조에 대한 스키마 파일도 이 폴더에 있습니다.

Flex SDK 설정

Adobe® Flex™에서 Adobe® AIR® 응용 프로그램을 개발하기 위해서는 다음과 같은 옵션이 있습니다.

- Adobe AIR 프로젝트를 만들고 AIR 응용 프로그램을 테스트, 디버깅 및 패키지화하기 위한 통합 도구를 제공하는 Adobe® Flash® Builder™를 다운로드하고 설치할 수 있습니다. 18페이지의 “Flash Builder에서 첫 번째 데스크톱 Flex AIR 응용 프로그램 만들기”를 참조하십시오.
- Adobe® Flex™ SDK를 다운로드하고 자주 사용하는 텍스트 편집기 및 명령줄 도구를 사용하여 Flex AIR 응용 프로그램을 개발할 수 있습니다.

Flex SDK로 AIR 응용 프로그램을 만드는 방법에 대한 간단한 개요를 보려면 34페이지의 “Flex SDK를 사용하여 첫 번째 데스크톱 AIR 응용 프로그램 만들기”를 참조하십시오.

Flex SDK 설치

명령줄 도구를 사용하여 AIR 응용 프로그램을 만들려면 컴퓨터에 Java가 설치되어 있어야 합니다. JRE 또는 JDK(버전 1.5 이상)에서 Java 가상 시스템을 사용할 수 있습니다. Java JRE 및 JDK는 http://java.sun.com/에서 제공됩니다.

참고: 최종 사용자가 AIR 응용 프로그램을 실행할 때는 Java가 필요하지 않습니다.

Flex SDK는 AIR 응용 프로그램을 패키지, 컴파일 및 디버그하는 데 사용하는 AIR API 및 명령줄 도구를 제공합니다.

1. 아직 Flex SDK를 설치하지 않은 경우 http://opensource.adobe.com/wiki/display/flexsdk/Downloads에서 다운로드하십시오.
2. SDK의 내용을 폴더(예: Flex SDK)에 저장합니다.
3. AIR SDK의 내용을 Flex SDK에 있는 파일에 복사합니다.

   참고: Mac 컴퓨터에서는 전체 디렉토리가 아니라 SDK 폴더에서 개별 파일을 복사하거나 바꿔야 합니다. 기본적으로 Mac에 있는 디렉토리를 같은 이름의 디렉토리로 복사하면 대상 디렉토리에 있는 기존 파일이 제거됩니다. 즉, 두 디렉토리의 내용이 병합되지 않습니다. 터미널 윈도우에서 ditto 명령을 사용하여 AIR SDK를 Flex SDK:ditto air_sdk_folder flex_sdk_folder에 병합할 수 있습니다.
4. 명령줄 AIR 유틸리티는 bin 폴더에 있습니다.

외부 SDK 설정

Android 및 iOS용 응용 프로그램을 개발하려면 플랫폼 제조업체로부터 프로비저닝 파일, SDK 또는 기타 개발 도구를 다운로드해야 합니다.

Android SDK 다운로드 및 설치에 대한 자세한 내용은 Android 개발자: SDK 설치를 참조하십시오. AIR 2.6부터는 Android SDK를 다운로드하지 않아도 됩니다. 이제는 APK 패키지를 설치하고 실행하는 데 필요한 기본 구성 요소가 AIR SDK에 포함되어 있기 때문입니다. 하지만 Android SDK는 소프트웨어 에뮬레이터를 만들고 실행하거나 장치 스크린 샷을 생성하는 등의 다양한 개발 작업에 유용합니다.

iOS 개발에는 외부 SDK가 필요하지 않습니다. 하지만 특수 인증서 및 프로비저닝 프로파일이 필요합니다. 자세한 내용은 Apple로부터 개발자 파일 얻기를 참조하십시오.

5장: 첫 번째 AIR 응용 프로그램 만들기

Flash Builder에서 첫 번째 데스크톱 Flex AIR 응용 프로그램 만들기

Adobe® AIR® 작동 방식을 빠르게 살펴보려면 다음 지침에 따라 Adobe® Flash® Builder를 사용하여 간단한 SWF 파일 기반 AIR "Hello World" 응용 프로그램을 만들어 패키지화해 보십시오.

아직 Flash Builder를 설치하지 않았다면 지금 다운로드하여 설치하십시오. 또한 www.adobe.com/go/air_kr에서 최신 버전의 Adobe AIR를 다운로드하여 설치하십시오.

AIR 프로젝트 만들기

Flash Builder에는 AIR 응용 프로그램을 개발하고 패키지화할 수 있는 도구가 포함되어 있습니다.

다른 Flex 기반 응용 프로그램 프로젝트를 만들 때와 마찬가지로 새 프로젝트를 정의하여 Flash Builder 또는 Flex Builder에서 AIR 응용 프로그램을 만들 수 있습니다.

1 Flash Builder를 엽니다.

2 [File] > [New] > [Flex Project]를 선택합니다.

3 프로젝트 이름을 AIRHelloWorld로 입력합니다.

4 Flex에서 AIR 응용 프로그램은 응용 프로그램 유형으로 간주됩니다. 두 가지 방법이 있습니다.

- Adobe® Flash® Player에서 실행되는 웹 응용 프로그램
- Adobe AIR에서 실행되는 데스크톱 응용 프로그램

응용 프로그램 유형으로 [Desktop]을 선택합니다.

5 [Finish]를 클릭하여 프로젝트를 만듭니다.

초기 AIR 프로젝트는 기본 MXML 파일과 응용 프로그램 XML 파일(응용 프로그램 설명자 파일이라고 함)의 두 가지 파일로 구성됩니다. 후자는 응용 프로그램 속성을 지정합니다.

자세한 내용은 Flash Builder을 사용하여 AIR 응용 프로그램 개발을 참조하십시오.

AIR 응용 프로그램 코드 작성

"Hello World" 응용 프로그램 코드를 작성하려면 응용 프로그램 MXML 파일(AIRHelloWorld.mxml)을 편집기에서 열어 편집해야 합니다. 파일이 열려 있지 않으면 Project Navigator를 사용하여 여십시오.

데스크톱의 Flex AIR 응용 프로그램은 MXML WindowedApplication 태그 내에 포함됩니다. MXML WindowedApplication 태그는 제목 표시줄, 닫기 버튼 같은 기본 윈도우 컨트롤이 포함된 간단한 윈도우를 만듭니다.

1 WindowedApplication 구성 요소에 title 특성을 추가하고 값으로 "Hello World"를 할당합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                       xmlns:s="library://ns.adobe.com/flex/spark"
                       xmlns:mx="library://ns.adobe.com/flex/mx"
                       title="Hello World">
</s:WindowedApplication>
```

2 응용 프로그램에 Label 구성 요소를 추가(WindowedApplication 태그 안에 배치)합니다. Label 구성 요소의 text 속성을 "Hello AIR"로 설정한 후 다음과 같이 텍스트를 가운데에 맞추는 레이아웃 제한을 설정합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                       xmlns:s="library://ns.adobe.com/flex/spark"
                       xmlns:mx="library://ns.adobe.com/flex/mx"
                       title="Hello World">

    <s:Label text="Hello AIR" horizontalCenter="0" verticalCenter="0"/>
</s:WindowedApplication>
```

3 여는 WindowedApplication 태그와 앞서 입력한 레이블 구성 요소 태그 사이에 다음 스타일 블록을 추가합니다.

```
<fx:Style>
    @namespace s "library://ns.adobe.com/flex/spark";
    s|WindowedApplication
    {

skinClass:ClassReference("spark.skins.spark.SparkChromeWindowedApplicationSkin");
        background-color:#999999;
        background-alpha:"0.7";
    }
</fx:Style>
```

이 스타일 설정은 전체 응용 프로그램에 적용되어 윈도우 배경을 조금 투명한 회색으로 렌더링합니다.

이제 응용 프로그램 코드가 다음과 같아야 합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                       xmlns:s="library://ns.adobe.com/flex/spark"
                       xmlns:mx="library://ns.adobe.com/flex/mx"
                       title="Hello World">

    <fx:Style>
        @namespace s "library://ns.adobe.com/flex/spark";
        s|WindowedApplication
        {

skinClass:ClassReference("spark.skins.spark.SparkChromeWindowedApplicationSkin");
            background-color:#999999;
            background-alpha:"0.7";
        }
    </fx:Style>

    <s:Label text="Hello AIR" horizontalCenter="0" verticalCenter="0"/>
</s:WindowedApplication>
```

다음으로, 응용 프로그램 설명자에서 응용 프로그램을 투명하게 만드는 몇 가지 설정을 변경합니다.

1 Flex Navigator 윈도우에서 프로젝트의 소스 디렉토리에 있는 응용 프로그램 설명자 파일을 찾습니다. 프로젝트 이름을 AIRHelloWorld로 지정했으므로 이 파일의 이름은 AIRHelloWorld-app.xml입니다.

2 응용 프로그램 설명자 파일을 두 번 클릭하여 Flash Builder에서 편집합니다.

3 XML 코드에서 systemChrome 및 transparent 속성에 대한 주석 줄을 찾습니다. 이 줄은 initialWindow 속성 안에 있습니다. 주석을 제거합니다. 즉, "<!--" 및 "-->" 주석 구분 기호를 제거하십시오.

4 다음과 같이 systemChrome 속성의 텍스트 값을 none으로 설정합니다.

```
<systemChrome>none</systemChrome>
```

5 다음과 같이 transparent 속성의 텍스트 값을 true로 설정합니다.

```
<transparent>true</transparent>
```

6 파일을 저장합니다.

AIR 응용 프로그램 테스트

작성한 응용 프로그램 코드를 테스트하려면 디버그 모드에서 응용 프로그램을 실행합니다.

1 기본 툴바에서 [Debug] 버튼 을 클릭합니다.

또는 [Run] > [Debug] > [AIRHelloWorld] 명령을 선택할 수도 있습니다.

최종 AIR 응용 프로그램은 다음 예제와 같은 모습입니다.

2 Label 컨트롤의 horizontalCenter 및 verticalCenter 속성을 사용하여 텍스트를 윈도우 가운데에 배치합니다. 다른 모든 데스크톱 응용 프로그램과 마찬가지로 윈도우를 이동하거나 윈도우의 크기를 조정할 수 있습니다.

참고: 응용 프로그램이 컴파일되지 않으면 코드에서 잘못 입력한 구문 오류나 맞춤법 오류가 없는지 확인하고 해당 오류를 수정합니다. 오류와 경고는 Flash Builder의 Problems 보기에 표시됩니다.

AIR 응용 프로그램 패키지, 서명 및 실행

이제 "Hello World" 응용 프로그램을 AIR 파일로 패키지화하여 배포할 수 있습니다. AIR 파일은 프로젝트의 bin 폴더에 들어 있는 모든 파일(응용 프로그램 파일)이 포함된 보관 파일입니다. 이 간단한 예제에서는 SWF 파일과 응용 프로그램 XML 파일이 보관 파일에 포함됩니다. 이제 AIR 패키지를 배포하면 사용자가 응용 프로그램을 설치하고 사용할 수 있습니다. 이 과정의 필수 단계는 패키지에 디지털 서명을 하는 것입니다.

1 응용 프로그램에서 컴파일 오류가 발생하지 않고 예상대로 실행되는지 확인합니다.

2 [프로젝트] > [릴리스 빌드 내보내기]를 선택합니다.

3 AIRHelloWorld 프로젝트 및 AIRHelloWorld.mxml 응용 프로그램이 프로젝트 및 응용 프로그램의 목록에 있는지 확인합니다.

4 [Export as signed AIR package] 옵션을 선택합니다. 그런 다음 [Next]를 클릭합니다.

5 사용할 수 있는 기존 디지털 인증서가 있다면 [Browse]를 클릭하여 인증서를 선택합니다.

6 새 자체 서명 디지털 인증서를 만들려면 [Create]를 선택합니다.

7 필요한 정보를 입력하고 [OK]를 클릭합니다.

8 [Finish]를 클릭하여 AIR 패키지를 생성합니다. 생성된 패키지의 이름은 AIRHelloWorld.air입니다.

이제 Flash Builder의 Project Navigator나 파일 시스템에서 AIR 파일을 두 번 클릭하여 응용 프로그램을 설치하고 실행할 수 있습니다.

Flash Professional을 사용하여 첫 번째 데스크톱 AIR 응용 프로그램 만들기

이 장의 지시에 따라 Adobe® Flash® Professional을 사용하여 간단한 "Hello World" AIR 응용 프로그램을 직접 만들고 패키지화하면 Adobe® AIR®의 작동 방식을 빠르고 쉽게 확인해 볼 수 있습니다.

아직 Adobe AIR를 설치하지 않은 경우 www.adobe.com/go/air_kr에서 다운로드하여 설치하십시오.

Flash에서 Hello World 응용 프로그램 만들기

Flash에서 Adobe AIR 응용 프로그램을 만드는 것은 다른 FLA 파일을 만드는 것과 비슷합니다. 다음 절차를 따라 Flash Professional로 간단한 Hello World 응용 프로그램을 만들어 보십시오.

Hello World 응용 프로그램을 만들려면

1. Flash를 시작합니다.
2. 시작 화면에서 AIR를 클릭하고 Adobe AIR 제작 설정을 지정하여 빈 FLA 파일을 만듭니다.
3. [도구] 패널에서 [텍스트] 도구를 선택하여 스테이지 가운데에 정적 텍스트 필드(기본값)를 만듭니다. 15-20자가 들어갈 정도로 넓게 만드십시오.
4. 텍스트 필드에 "Hello World"를 입력합니다.
5. 파일을 저장하고 이름(예: HelloAIR)을 지정합니다.

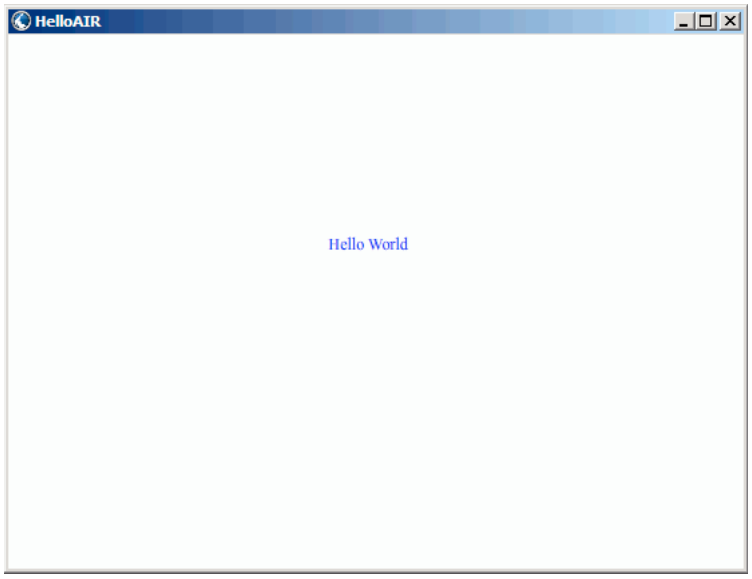
응용 프로그램 테스트

1. Ctrl+Enter를 누르거나 [컨트롤] -> [동영상 테스트] -> [테스트]를 선택하여 Adobe AIR에서 응용 프로그램을 테스트합니다.
2. [동영상 디버그] 기능을 사용하려면 먼저 응용 프로그램에 ActionScript 코드를 추가해야 합니다. 다음과 같은 trace 문을 간단히 추가해 볼 수 있습니다.

   ```
   trace("Running AIR application using Debug Movie");
   ```

3. Ctrl+Shift+Enter를 누르거나 [디버그] -> [동영상 디버그] -> [디버그]를 선택하여 [동영상 디버그]로 응용 프로그램을 실행합니다.

Hello World 응용 프로그램은 다음 그림과 같습니다.



응용 프로그램 패키지화

1 [파일] > [제작]을 선택합니다.

2 기존 디지털 인증서로 Adobe AIR 패키지를 서명하거나, 다음 단계를 따라 자체 서명 인증서를 만듭니다.

   a [인증서] 필드 옆의 [새로 만들기] 버튼을 클릭합니다.

   b [제작자 이름], [조직 구성 단위], [조직 이름], [전자 메일], [국가], [암호] 및 [암호 확인] 필드에 값을 입력합니다.

   c 인증서 유형을 지정합니다. 인증서 [유형] 옵션은 보안 수준을 가리킵니다. [1024-RSA]는 1024비트 키(보안 수준 낮음)를 사용하고 [2048-RSA]는 2048비트 키(보안 수준 높음)를 사용합니다.

   d [다른 이름으로 저장] 필드에 값을 입력하거나 [찾아보기...] 버튼을 클릭하고 폴더 위치를 찾아 인증서 파일의 정보를 저장합니다(예: **C:/Temp/mycert.pfx**). 작업을 마치면 [확인]을 클릭합니다.

   e [디지털 서명] 대화 상자가 다시 나타납니다. 만든 자체 서명된 인증서의 경로 및 파일 이름이 [인증서] 텍스트 상자에 표시됩니다. 표시되지 않는 경우, 경로 및 파일 이름을 입력하거나 [찾아보기] 버튼을 클릭하여 파일을 찾아 선택합니다.

   f b단계에서 지정한 암호와 동일한 암호를 [디지털 서명] 대화 상자의 [암호] 텍스트 필드에 입력합니다. Adobe AIR 응용 프로그램 서명에 대한 자세한 내용은 170페이지의 “AIR 파일에 디지털 서명”을 참조하십시오.

3 응용 프로그램 및 설치 프로그램 파일을 만들려면 [제작] 버튼을 클릭합니다. Flash CS4 및 CS5의 경우 [확인] 버튼을 클릭합니다. AIR 파일을 만들기 전에 [동영상 테스트] 또는 [동영상 디버그]를 실행하여 SWF 파일 및 application.xml 파일을 만들어야 합니다.

4 응용 프로그램을 설치하려면 응용 프로그램을 저장한 폴더에서 AIR 파일(**application**.air)을 두 번 클릭합니다.

5 [응용 프로그램 설치] 대화 상자에서 [설치] 버튼을 클릭합니다.

6 [설치 환경 설정] 및 [위치] 설정을 검토하고 '설치 후 응용 프로그램 시작' 체크 상자를 선택합니다. 그런 다음 [계속]을 클릭합니다.

7 '설치가 완료되었습니다.' 메시지가 나타나면 [완료]를 클릭합니다.

Flash Professional에서 첫 번째 Android용 AIR 응용 프로그램 만들기

Android용 AIR 응용 프로그램을 개발하려면 Android용 Flash Professional CS5 확장을 Adobe Labs에서 다운로드해야 합니다.

또한 Android 개발자: SDK 설치에 설명된 대로 Android 웹 사이트에서 Android SDK를 다운로드하여 설치해야 합니다.

프로젝트 만들기

1 Flash Professional CS5를 엽니다.

2 새로운 AIR for Android 프로젝트를 만듭니다.

Flash Professional 홈 화면에 AIR for Android 응용 프로그램을 만들기 위한 링크가 있습니다. [파일] > [새로 만들기]를 선택하여 AIR for Android 템플릿을 선택할 수도 있습니다.

3 문서를 HelloWorld.fla로 저장합니다.

코드 작성

이 자습서는 코드 작성을 위한 것이 아니기 때문에 간단히 텍스트 도구를 사용하여 스테이지에 "Hello, World!"라고만 씁니다.

응용 프로그램 속성 설정

1 [파일] > [AIR Android 설정]을 선택합니다.

2 [일반] 탭에서 다음 설정을 적용합니다.

- 출력 파일: HelloWorld.apk
- 응용 프로그램 이름: HelloWorld
- 응용 프로그램 ID: HelloWorld
- 버전 번호: 0.0.1
- 종횡비: 세로

3 [배포] 탭에서 다음 설정을 적용합니다.

- 인증서: 유효한 AIR 코드 서명 인증서를 가리킵니다. [생성] 버튼을 클릭하여 새 인증서를 만들 수 있습니다. Android Marketplace를 통해 배포된 Android 응용 프로그램에는 최소한 2033년까지 유효한 인증서가 있어야 합니다. [암호] 필드에 인증서 암호를 입력합니다.
- Android 배포 유형: 디버그
- 제작 후: 두 옵션 모두 선택
- Android SDK 디렉토리의 tools 하위 디렉토리에 있는 ADB 도구의 경로를 입력합니다.

4 [확인]을 클릭하여 [Android 설정] 대화 상자를 닫습니다.

이 배포 단계에서는 응용 프로그램에 아이콘이나 권한이 필요하지 않습니다. 대부분의 Android용 AIR 응용 프로그램은 보호된 기능에 액세스하는 데 몇 가지 권한이 필요합니다. 너무 많은 권한을 요구하는 경우 사용자가 응용 프로그램을 거부할 수도 있기 때문에 꼭 필요한 권한만 설정해야 합니다.

5 파일을 저장합니다.

Android 장치에서 응용 프로그램 패키지화 및 설치

1 장치에서 USB 디버깅이 사용되고 있는지 확인합니다. [응용 프로그램] > [배포] 아래의 설정 응용 프로그램에서 USB 디버깅을 켤 수 있습니다.

2 USB 케이블을 사용하여 장치를 컴퓨터에 연결합니다.

3 아직 AIR 런타임을 설치하지 않은 경우 Android Market으로 가서 Adobe AIR를 다운로드하여 설치합니다. 159페이지의 “ADT installRuntime 명령”을 사용하여 AIR를 로컬에서 설치할 수도 있습니다. AIR SDK에는 Android 장치 및 에뮬레이터에서 사용할 Android 패키지가 포함되어 있습니다.

4 [파일] > [제작]을 선택합니다.

Flash Professional이 APK 파일을 만들고, 연결된 Android 장치에 응용 프로그램을 설치하고, 응용 프로그램을 실행합니다.

첫 번째 iOS용 AIR 응용 프로그램 만들기

AIR 2.6 이상, iOS 4.2 이상

Adobe 도구만 사용하여 iOS 응용 프로그램의 기본 기능을 코딩하고 구성하고 테스트할 수 있습니다. 하지만 장치에서 iOS 응용 프로그램을 설치하고 해당 응용 프로그램을 배포하려면 Apple iOS Developer 프로그램(유료 서비스)에 가입해야 합니다. iOS Developer 프로그램에 가입하면 iOS Provisioning Portal에 액세스하여 테스트 및 후속 배포를 위해 응용 프로그램을 장치에 설치하는 데 필요한 항목 및 파일을 얻을 수 있습니다. Apple에서 제공하는 이들 항목 및 파일은 다음과 같습니다.

- 개발 및 배포 인증서
- 응용 프로그램 ID
- 개발 및 배포 프로비저닝 파일

응용 프로그램 내용 만들기

“Hello world!”라는 텍스트를 표시하는 SWF 파일을 만들어 보십시오. Flash Professional, Flash Builder 또는 다른 IDE를 사용하여 이 작업을 수행할 수 있습니다. 이 예제에서는 간단히 텍스트 편집기 그리고 Flex SDK에 포함된 명령줄 SWF 컴파일러를 사용합니다.

1 편리한 위치에 응용 프로그램 파일을 저장하기 위한 디렉토리를 만듭니다. **HelloWorld.as**라는 파일을 만들고 선호하는 코드 편집기에서 이 파일을 편집합니다.

2 다음 코드를 추가합니다.

```
package{

    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.text.TextFormat;
    import flash.text.TextFieldAutoSize;

    public class HelloWorld extends Sprite
    {
        public function HelloWorld():void
        {
            var textField:TextField = new TextField();
            textField.text = "Hello World!";
            textField.autoSize = TextFieldAutoSize.LEFT;

            var format:TextFormat = new TextFormat();
            format.size = 48;

            textField.setTextFormat ( format );
            this.addChild( textField );
        }
    }
}
```

3 amxmlc 컴파일러를 사용하여 클래스를 컴파일합니다.

```
amxmlc HelloWorld.as
```

같은 폴더에 **HelloWorld.swf**라는 SWF 파일이 생성됩니다.

참고: 이 예제에서는 amxmlc가 들어 있는 디렉토리를 포함하도록 환경 경로 변수를 설정했다고 가정합니다. 경로 설정에 대한 자세한 내용은 277페이지의 “path 환경 변수”를 참조하십시오. 또는 amxmlc 그리고 이 예제에서 사용된 다른 명령줄 도구에 대한 전체 경로를 입력할 수 있습니다.

응용 프로그램의 아이콘 아트 및 초기 화면 아트 제작

모든 iOS 응용 프로그램에는 iTunes 응용 프로그램 사용자 인터페이스와 장치 화면에 표시되는 아이콘이 있습니다.

1 프로젝트 디렉토리에 새 디렉토리를 만들고 이름을 icons로 지정합니다.

2 icons 디렉토리에 PNG 파일을 3개 만들고 각각 Icon_29.png, Icon_57.png, Icon_512.png로 이름을 지정합니다.

3 PNG 파일을 편집하여 응용 프로그램에 적합한 아트를 만듭니다. 이 파일은 29 x 29 픽셀, 57 x 57 픽셀, 512 x 512 픽셀 크기여야 합니다. 이 테스트에는 단색 사각형을 아트로 사용하면 됩니다.

참고: Apple App Store로 응용 프로그램을 전송할 때는 PNG 버전이 아니라 JPG 버전의 512픽셀 파일을 사용합니다. PNG 버전은 응용 프로그램의 개발 버전을 테스트하는 동안에만 사용합니다.

모든 iPhone 응용 프로그램은 iPhone에서 로드되는 동안 초기 이미지를 표시합니다. 이 초기 이미지는 PNG 파일로 정의합니다.

1 기본 개발 디렉토리에 Default.png라는 PNG 파일을 만듭니다. 이 파일을 icons 하위 디렉토리에 넣지 마십시오. 파일 이름은 반드시 대문자 D로 시작하는 Default.png이어야 합니다.

2 폭 320픽셀, 높이 480 픽셀로 파일을 편집합니다. 지금은 내용을 흰색 사각형으로 두면 됩니다. 내용은 나중에 변경합니다.

이 그래픽에 대한 자세한 내용은 79페이지의 “응용 프로그램 아이콘”을 참조하십시오.

응용 프로그램 설명자 파일 만들기

응용 프로그램의 기본 속성을 지정하는 응용 프로그램 설명자 파일을 만드십시오. Flash Builder 등의 IDE 또는 텍스트 편집기를 사용하여 이 작업을 완료할 수 있습니다.

1 HelloWorld.as가 들어 있는 프로젝트 폴더에서 **HelloWorld-app.xml**이라는 XML 파일을 만듭니다. 선호하는 XML 편집기에서 이 파일을 편집합니다.

2 다음 XML을 추가합니다.

```
<?xml version="1.0" encoding="utf-8" ?>
<application xmlns="http://ns.adobe.com/air/application/2.7" minimumPatchLevel="0">
    <id>change_to_your_id</id>
    <name>Hello World iOS</name>
    <versionNumber>0.0.1</versionNumber>
    <filename>HelloWorld</filename>
    <supportedProfiles>mobileDevice</supportedProfiles>
    <initialWindow>
        <content>HelloWorld.swf</content>
        <title>Hello World!</title>
    </initialWindow>
    <icon>
        <image29x29>icons/AIRApp_29.png</image29x29>
        <image57x57>icons/AIRApp_57.png</image57x57>
        <image512x512>icons/AIRApp_512.png</image512x512>
    </icon>
</application>
```

단순하게 만들기 위해 이 예제에서는 사용 가능한 속성 중 몇 개만 설정합니다.

참고: AIR 2 이하를 사용하는 경우 <versionNumber> 요소 대신 <version> 요소를 사용해야 합니다.

3 iOS Provisioning Portal에서 지정한 응용 프로그램 ID와 일치하도록 응용 프로그램 ID를 변경합니다. ID 앞에 임의의 번들 시드 부분을 포함시키지 마십시오.

4 ADL을 사용하여 응용 프로그램을 테스트합니다.

```
adl HelloWorld-app.xml -screensize iPhone
```

ADL은 **Hello World!**라는 텍스트가 표시되는 윈도우를 데스크톱에서 열어야 합니다. 그렇지 못하면 소스 코드 및 응용 프로그램 설명자에 오류가 있는지 확인해 보십시오.

IPA 파일 컴파일

이제 ADT를 사용하여 IPA 설치 프로그램 파일을 컴파일할 수 있습니다. P12 파일 포맷의 Apple 개발자 인증서 및 개인 키 그리고 Apple 개발 프로비저닝 프로파일이 있어야 합니다.

다음 옵션을 사용하여 ADT 유틸리티를 실행하십시오. 이때 keystore, storepass 및 provisioning-profile 값은 상황에 맞게 바꾸십시오.

```
adt -package -target ipa-debug
    -keystore iosPrivateKey.p12 -storetype pkcs12 -storepass qwerty12
    -provisioning-profile ios.mobileprovision
    HelloWorld.ipa
    HelloWorld-app.xml
    HelloWorld.swf icons Default.png
```

명령줄을 하나만 사용하십시오. 이 예제의 행 분리는 읽기 쉽게 만들기 위해 추가된 것일 뿐입니다.

ADT가 프로젝트 디렉토리에 **HelloWorld.ipa**라는 iOS 응용 프로그램 설치 프로그램 파일을 생성합니다. IPA 파일을 컴파일하는 데는 몇 분 정도 걸릴 수 있습니다.

장치에 응용 프로그램 설치

테스트를 위해 iOS 응용 프로그램을 설치하려면

1 iTunes 응용 프로그램을 엽니다.

2 이 응용 프로그램의 프로비저닝 프로파일을 iTunes에 추가합니다. iTunes에서 [파일] > [보관함에 추가]를 선택합니다. 그런 다음 프로비저닝 프로파일 파일(파일 유형이 mobileprovision인 파일)을 선택합니다.

   지금은 개발자 장치에서 응용 프로그램을 테스트하는 것이므로 개발용 프로비저닝 프로파일을 사용합니다.

   나중에 iTunes Store에 응용 프로그램을 배포할 때는 배포용 프로파일을 사용해야 합니다. 애드혹 방식으로 응용 프로그램을 배포(iTunes Store를 통하지 않고 여러 장치에 배포)하려면 애드혹용 프로비저닝 프로파일을 사용합니다.

   프로비저닝 프로파일에 대한 자세한 내용은 60페이지의 “iOS 설정”을 참조하십시오.

3 일부 iTunes 버전은 같은 버전의 응용 프로그램이 이미 설치되어 있으면 응용 프로그램을 바꾸지 않습니다. 이 경우 장치와 iTunes의 응용 프로그램 목록에서 응용 프로그램을 삭제하십시오.

4 응용 프로그램의 IPA 파일을 두 번 클릭합니다. 해당 파일이 iTunes에서 응용 프로그램 목록에 나타납니다.

5 장치를 컴퓨터의 USB 포트에 연결합니다.

6 iTunes에서 장치의 [응용 프로그램] 탭에 있는 설치할 응용 프로그램 목록에 응용 프로그램이 선택되어 있는지 확인합니다.

7 iTunes 응용 프로그램의 왼쪽 목록에서 장치를 선택합니다. 그런 다음 [동기화] 버튼을 클릭합니다. 동기화가 완료되면 iPhone에 Hello World 응용 프로그램이 나타납니다.

새 버전이 설치되지 않은 경우 장치와 iTunes의 응용 프로그램 목록에서 응용 프로그램을 삭제하고 이 절차를 다시 실행합니다. 현재 설치된 버전의 응용 프로그램 ID와 버전이 동일하기 때문일 수 있습니다.

초기 화면 그래픽 편집

앞서 응용 프로그램을 컴파일하기 전에 Default.png 파일(25페이지의 “응용 프로그램의 아이콘 아트 및 초기 화면 아트 제작” 참조)을 만들었습니다. 이 PNG 파일은 응용 프로그램이 로드되는 동안 시작 이미지로 사용됩니다. iPhone에서 응용 프로그램을 테스트하는 경우 시작 시에 이 빈 화면이 표시되는 것을 볼 수 있습니다.

이 이미지를 응용 프로그램의 시작 화면("Hello World!")과 일치하도록 변경해야 합니다.:

1 장치에서 응용 프로그램을 엽니다. 첫 번째 "Hello World" 텍스트가 나타나면 화면 아래쪽에 있는 [홈] 버튼을 누르고 있습니다. [홈] 버튼을 누른 상태에서 iPhone 맨 위에 있는 [잠자기/깨우기] 버튼을 누릅니다. 이렇게 하면 스크린 샷이 생성되어 카메라 롤로 보내집니다.

2 iPhoto 또는 기타 사진 전송 응용 프로그램에서 개발 컴퓨터로 이미지를 전송합니다. Mac OS에서는 이미지 캡처 응용 프로그램을 사용할 수도 있습니다.

   전자 메일을 통해 사진을 개발 컴퓨터로 보낼 수도 있습니다.

   - Photos 응용 프로그램을 엽니다.
   - 카메라 롤을 엽니다.
   - 캡처한 스크린 샷 이미지를 엽니다.
   - 이미지를 누르고 왼쪽 아래에 있는 "전달"(화살표) 버튼을 누릅니다. 그런 다음 [이메일 메시지에 사진 보내기] 버튼을 눌러 이미지를 자신의 전자 메일 주소로 보냅니다.

3 개발 디렉토리에 있는 Default.png 파일을 화면 캡처 이미지의 PNG 버전으로 바꿉니다.

4 응용 프로그램을 다시 컴파일(26페이지의 “IPA 파일 컴파일” 참조)하고 장치에 다시 설치합니다.

이제 응용 프로그램을 로드할 때 새 시작 화면이 사용됩니다.

참고: 크기(320 x 480 픽셀)만 맞으면 어떤 아트든 Default.png 파일로 만들 수 있습니다. 그러나 일반적으로 Default.png 이미지는 응용 프로그램의 초기 상태와 일치하는 것이 좋습니다.

Dreamweaver를 사용하여 첫 번째 HTML 기반 AIR 응용 프로그램 만들기

Adobe® AIR® 작동 방식을 빠르게 살펴보려면 다음 지침에 따라 Adobe® AIR® Extension for Dreamweaver®를 사용하여 간단한 HTML 기반 AIR "Hello World" 응용 프로그램을 만들고 패키지화해 보십시오.

아직 Adobe AIR를 설치하지 않은 경우 www.adobe.com/go/air_kr에서 다운로드하여 설치하십시오.

Adobe AIR Extension for Dreamweaver를 설치하는 방법은 Adobe AIR Extension for Dreamweaver 설치를 참조하십시오.

시스템 요구 사항을 비롯한 Extension에 대한 개요는 AIR Extension for Dreamweaver를 참조하십시오.

참고: HTML 기반 AIR 응용 프로그램은 데스크톱 및 extendedDesktop 프로파일에 대해서만 개발할 수 있습니다. 모바일 프로파일은 지원되지 않습니다.

응용 프로그램 파일 준비

Adobe AIR 응용 프로그램의 시작 페이지와 모든 관련 페이지가 Dreamweaver 사이트에 정의되어 있어야 합니다.

1. Dreamweaver를 시작하고 사이트를 정의했는지 확인합니다.
2. [파일] > [새로 만들기]를 선택한 다음 [페이지 유형] 열에서 HTML을 선택하고 [레이아웃] 열에서 [없음]을 선택한 후 [만들기]를 클릭하여 새 HTML 페이지를 엽니다.
3. 새 페이지에서 **Hello World!**를 입력합니다.

   이 예제는 매우 단순합니다. 원한다면 텍스트에 좋아하는 스타일을 지정하고, 페이지에 더 많은 내용을 추가하고, 다른 페이지를 이 시작 페이지에 링크할 수 있습니다.
4. [파일] > [저장]을 차례로 클릭한 다음 페이지를 hello_world.html로 저장합니다. 파일을 Dreamweaver 사이트에 저장했는지 확인합니다.

Dreamweaver 사이트에 대한 자세한 내용은 Dreamweaver 도움말을 참조하십시오.

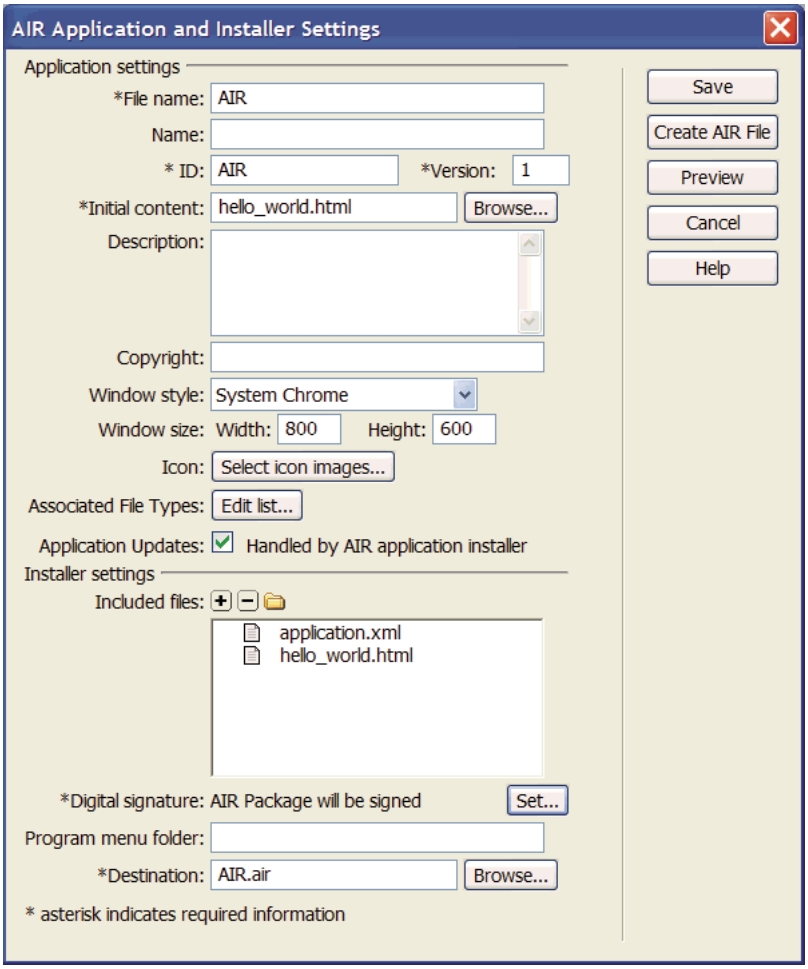
Adobe AIR 응용 프로그램 만들기

1. hello_world.html 페이지가 Dreamweaver 문서 윈도우에 열려 있어야 합니다. 만드는 방법은 이전 단원을 참조하십시오.
2. [사이트] > [Air 응용 프로그램 설정]을 선택합니다.

   [AIR 응용 프로그램 및 설정] 대화 상자에 있는 대부분의 필수 설정은 자동으로 입력됩니다. 하지만 응용 프로그램의 초기 내용(또는 시작 페이지)은 직접 선택해야 합니다.
3. [초기 내용] 옵션 옆에 있는 [찾아보기] 버튼을 클릭하고 hello_world.html 페이지를 찾아 선택합니다.
4. [디지털 서명] 옵션 옆에 있는 [설정] 버튼을 클릭합니다.

   디지털 서명은 소프트웨어 작성자가 만든 이후로 응용 프로그램의 코드가 변경되거나 손상되지 않았음을 보장하며 모든 Adobe AIR 응용 프로그램에 필요합니다.
5. [디지털 서명] 대화 상자에서 [디지털 인증서로 AIR 패키지 서명]을 선택한 다음 [만들기] 버튼을 클릭합니다. 디지털 인증서에 액세스할 수 있는 경우에는 [찾아보기] 버튼을 클릭하여 선택할 수도 있습니다.
6. [자체 서명된 디지털 인증서] 대화 상자의 필수 필드에 필요한 정보를 모두 입력합니다. 이름, 암호, 암호 확인을 입력한 다음 디지털 인증서 파일의 이름을 입력해야 합니다. 디지털 인증서는 사이트 루트에 저장됩니다.
7. [확인]을 클릭하여 [디지털 서명] 대화 상자로 돌아갑니다.
8. [디지털 서명] 대화 상자에서 디지털 인증서에 지정한 암호를 입력하고 [확인]을 클릭합니다.

작업을 마치면 [AIR 응용 프로그램 및 설치 프로그램 설정] 대화 상자가 다음과 같이 표시됩니다.



모든 대화 상자 옵션 및 해당 옵션을 편집하는 방법에 대해서는 Dreamweaver에서 AIR 응용 프로그램 만들기를 참조하십시오.

9 [AIR 파일 만들기] 버튼을 클릭합니다.

Dreamweaver에서 Adobe AIR 응용 프로그램을 만들어 사이트 루트 폴더에 저장합니다. 또한 application.xml 파일을 만들어 같은 위치에 저장합니다. 이 파일은 응용 프로그램의 다양한 속성을 정의하는 매니페스트로 사용됩니다.

데스크톱에 응용 프로그램 설치

이제 응용 프로그램 파일을 만들었으므로 원하는 데스크톱에 설치할 수 있습니다.

1 Adobe AIR 응용 프로그램 파일을 Dreamweaver 사이트에서 자신의 데스크톱으로 이동하거나 다른 데스크톱으로 이동합니다.

이 단계는 선택 사항입니다. 원하는 경우 Dreamweaver 사이트 디렉토리에서 사용자 컴퓨터에 직접 새 응용 프로그램을 설치할 수 있습니다.

2 응용 프로그램 실행 파일(.air 파일)을 두 번 클릭하여 응용 프로그램을 설치합니다.

Adobe AIR 응용 프로그램 미리 보기

AIR 응용 프로그램의 일부가 되는 페이지는 언제라도 미리 볼 수 있습니다. 즉, 응용 프로그램을 설치했을 때 어떻게 보이는지 확인하기 위해 응용 프로그램을 패키지화할 필요가 없습니다.

1 Dreamweaver 문서 윈도우에 hello_world.html 페이지가 열려 있어야 합니다.

2 [문서] 툴바에서 [미리 보기]/[브라우저에서 디버그] 버튼을 클릭한 다음 [AIR에서 미리 보기]를 선택합니다.

Ctrl+Shift+F12(Windows) 또는 Cmd+Shift+F12(Macintosh)를 눌러 이 작업을 수행할 수도 있습니다.

이 페이지를 미리 보면 사용자가 데스크톱에 응용 프로그램을 설치한 후 응용 프로그램의 시작 페이지로 표시되는 내용을 확인할 수 있습니다.

AIR SDK를 사용하여 첫 번째 HTML 기반 AIR 응용 프로그램 만들기

Adobe® AIR® 작동 방식을 빠르게 살펴보려면 다음 지침에 따라 간단한 HTML 기반 AIR "Hello World" 응용 프로그램을 만들고 패키지 해보십시오.

시작하려면 런타임을 설치하고 AIR SDK를 설정해야 합니다. 이 자습서에서는 ADL(**AIR Debug Launcher**)과 ADT(**AIR Developer Tool**)를 사용합니다. ADL 및 ADT는 명령줄 유틸리티 프로그램이며 AIR SDK의 bin 디렉토리에서 찾을 수 있습니다(15페이지의 “AIR SDK 설치” 참조). 이 자습서에서는 학습자가 명령줄에서 프로그램을 실행하는 데 익숙하며 운영 체제에 필요한 Path 환경 변수를 설정하는 방법을 알고 있다고 가정합니다.

참고: Adobe® Dreamweaver® 사용자는 28페이지의 “Dreamweaver를 사용하여 첫 번째 HTML 기반 AIR 응용 프로그램 만들기”를 읽어보십시오.

참고: HTML 기반 AIR 응용 프로그램은 데스크톱 및 extendedDesktop 프로파일에 대해서만 개발할 수 있습니다. 모바일 프로파일은 지원되지 않습니다.

프로젝트 파일 만들기

모든 HTML 기반 AIR 프로젝트에는 응용 프로그램 메타데이터를 지정하는 응용 프로그램 설명자 파일과 최상위 HTML 페이지의 두 가지 파일이 포함되어야 합니다. 이러한 필수 파일과 함께 이 프로젝트에는 AIR API 클래스에서 편리하게 사용할 수 있는 별칭 변수를 정의하는 JavaScript 코드 파일인 AIRAliases.js가 포함되어 있습니다.

1 프로젝트 파일을 저장할 HelloWorld 디렉토리를 만듭니다.

2 HelloWorld-app.xml이라는 XML 파일을 만듭니다.

3 HelloWorld.html이라는 HTML 파일을 만듭니다.

4 AIR SDK의 frameworks 폴더에 있는 AIRAliases.js를 프로젝트 디렉토리에 복사합니다.

AIR 응용 프로그램 설명자 파일 만들기

AIR 응용 프로그램을 작성하려면 다음과 같은 구조로 XML 응용 프로그램 설명자 파일을 만들어야 합니다.

```
<application xmlns="...">
    <id>…</id>
    <versionNumber>…</versionNumber>
    <filename>…</filename>
    <initialWindow>
        <content>…</content>
        <visible>…</visible>
        <width>…</width>
        <height>…</height>
    </initialWindow>
</application>
```

1 HelloWorld-app.xml을 편집할 수 있도록 엽니다.

2 AIR 네임스페이스 특성을 포함하는 루트 <application> 요소를 추가합니다.

<application xmlns="http://ns.adobe.com/air/application/2.7"> 네임스페이스의 마지막 세그먼트인 "2.7"은 응용 프로그램에 필요한 런타임 버전을 지정합니다.

3 <id> 요소를 추가합니다.

<id>examples.html.HelloWorld</id> 응용 프로그램 ID는 제작자 ID(응용 프로그램 패키지에 서명하는 데 사용된 인증서에서 파생됨)와 함께 응용 프로그램을 고유하게 식별합니다. 응용 프로그램 ID는 설치, 개인 응용 프로그램 파일 시스템 저장소 디렉토리 액세스, 개인 암호화 저장소 액세스 및 응용 프로그램 간 통신에 사용됩니다.

4 <versionNumber> 요소를 추가합니다.

<versionNumber>0.1</versionNumber> 사용자에게 설치하는 응용 프로그램의 버전을 알려 줍니다.

참고: AIR 2 이하를 사용하는 경우 <versionNumber> 요소 대신 <version> 요소를 사용해야 합니다.

5 <filename> 요소를 추가합니다.

<filename>HelloWorld</filename> 응용 프로그램 실행 파일, 설치 디렉토리 및 운영 체제에서 응용 프로그램에 대한 기타 참조로 사용되는 이름입니다.

6 초기 응용 프로그램 윈도우의 속성을 지정하는 다음과 같은 자식 요소를 포함하는 <initialWindow> 요소를 추가합니다.

<content>HelloWorld.html</content> AIR에서 로드할 루트 HTML 파일을 식별합니다.

<visible>true</visible> 윈도우를 즉시 보이게 만듭니다.

<width>400</width> 윈도우 폭을 픽셀 단위로 설정합니다.

<height>200</height> 윈도우 높이를 설정합니다.

7 파일을 저장합니다. 완성된 응용 프로그램 설명자 파일은 다음과 같아야 합니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://ns.adobe.com/air/application/2.7">
    <id>examples.html.HelloWorld</id>
    <versionNumber>0.1</versionNumber>
    <filename>HelloWorld</filename>
    <initialWindow>
        <content>HelloWorld.html</content>
        <visible>true</visible>
        <width>400</width>
        <height>200</height>
    </initialWindow>
</application>
```

이 예제에서는 사용 가능한 응용 프로그램 속성 중 일부만 설정합니다. 윈도우 크롬, 윈도우 크기, 투명도, 기본 설치 디렉토리, 연결된 파일 유형, 응용 프로그램 아이콘 등을 지정할 수 있는 전체 응용 프로그램 속성을 보려면 183페이지의 “AIR 응용 프로그램 설명자 파일”을 참조하십시오.

응용 프로그램 HTML 페이지 만들기

AIR 응용 프로그램의 기본 파일로 사용할 간단한 HTML 페이지를 만들어야 합니다.

1 HelloWorld.html 파일을 편집할 수 있도록 엽니다. 다음 HTML 코드를 추가합니다.

```
<html>
<head>
    <title>Hello World</title>
</head>
<body onLoad="appLoad()">
    <h1>Hello World</h1>
</body>
</html>
```

2 HTML의 <head> 섹션에서 AIRAliases.js 파일을 가져옵니다.

```
<script src="AIRAliases.js" type="text/javascript"></script>
```

AIR가 HTML 윈도우 객체에 runtime이라는 속성을 정의합니다. 이 runtime 속성은 클래스의 정규화된 패키지 이름을 사용하여 기본 제공 AIR 클래스에 액세스하는 방법을 제공합니다. 예를 들어, AIR File 객체를 만들려면 JavaScript에 다음과 같은 문을 추가합니다.

```
var textFile = new runtime.flash.filesystem.File("app:/textfile.txt");
```

AIRAliases.js 파일은 자주 사용하는 AIR API를 편리하게 사용할 수 있도록 별칭을 정의합니다. AIRAliases.js를 사용하면 File 클래스에 대한 참조를 다음과 같이 줄일 수 있습니다.

```
var textFile = new air.File("app:/textfile.txt");
```

3 AIRAliases 스크립트 태그 아래에 onLoad 이벤트를 처리하는 JavaScript 함수가 포함된 다른 스크립트 태그를 추가합니다.

```
<script type="text/javascript">
function appLoad(){
    air.trace("Hello World");
}
</script>
```

appLoad() 함수는 단순히 air.trace() 함수를 호출합니다. 이제 ADL을 사용하여 응용 프로그램을 실행하면 명령 콘솔에 추적 메시지가 출력됩니다. Trace 문은 디버깅 시 매우 유용하게 사용할 수 있습니다.

4 파일을 저장합니다.

이제 HelloWorld.html 파일이 다음과 같아야 합니다.

```
<html>
<head>
    <title>Hello World</title>
    <script type="text/javascript" src="AIRAliases.js"></script>
    <script type="text/javascript">
        function appLoad(){
            air.trace("Hello World");
        }
    </script>
</head>
<body onLoad="appLoad()">
    <h1>Hello World</h1>
</body>
</html>
```

응용 프로그램 테스트

명령줄에서 응용 프로그램을 실행하고 테스트하려면 ADL(AIR Debug Launcher) 유틸리티를 사용합니다. ADL 실행 파일은 AIR SDK의 bin 디렉토리에서 찾을 수 있습니다. AIR SDK를 아직 설치하지 않은 경우 15페이지의 "AIR SDK 설치"를 참조하십시오.

1 명령 콘솔이나 셸을 엽니다. 이 프로젝트용으로 만든 디렉토리로 변경합니다.

2 다음 명령을 실행합니다.

```
adl HelloWorld-app.xml
```

응용 프로그램을 표시하는 AIR 윈도우가 열립니다. 또한 콘솔 윈도우에 air.trace() 호출의 결과 메시지가 표시됩니다.

자세한 내용은 183페이지의 "AIR 응용 프로그램 설명자 파일"을 참조하십시오.

AIR 설치 파일 만들기

응용 프로그램이 성공적으로 실행되면 ADT 유틸리티를 사용하여 응용 프로그램을 AIR 설치 파일로 패키지화할 수 있습니다. AIR 설치 파일은 사용자에게 응용 프로그램을 배포할 수 있도록 응용 프로그램의 모든 파일을 포함하는 보관 파일입니다. 패키지화된 AIR 파일을 설치하려면 먼저 Adobe AIR를 설치해야 합니다.

응용 프로그램 보안을 유지하기 위해 모든 AIR 설치 파일은 디지털 서명이 되어야 합니다. ADT나 다른 인증서 생성 도구에서는 개발 용도로 사용할 기본적인 자체 서명 인증서를 생성할 수 있습니다. 또한 VeriSign이나 Thawte 같은 기업 인증 기관에서 상업적인 코드로 서명하는 인증서를 구매할 수도 있습니다. 사용자가 자체 서명된 AIR 파일을 설치할 경우 설치 과정에서 제작자가 "알 수 없음"으로 표시됩니다. 이것은 자체 서명된 인증서가 AIR 파일이 만들어진 이후로 변경되지 않았다는 것만 보장하기 때문입니다. 악의적인 AIR 파일을 자체 서명하여 응용 프로그램으로 제공하는 것 자체를 막을 방법은 없습니다. 따라서, 공개적으로 배포되는 AIR 파일에는 검증 가능한 상용 인증서를 사용하는 것이 좋습니다. AIR 보안 문제에 대한 개요는 AIR 보안 (ActionScript 개발자용) 또는 AIR security(HTML 개발자용)을 참조하십시오.

자체 서명 인증서 및 키 쌍 생성

❖ 명령 프롬프트에서 다음 명령을 입력합니다. ADT 실행 파일은 AIR SDK의 bin 디렉토리에 있습니다.

```
adt -certificate -cn SelfSigned 1024-RSA sampleCert.pfx samplePassword
```

ADT가 인증서와 관련 개인 키가 들어 있는 **sampleCert.pfx**라는 키 저장소 파일을 생성합니다.

이 예제에서는 인증서에 설정할 수 있는 최소한의 특성만 사용합니다. 키 유형은 **1024-RSA** 또는 **2048-RSA**여야 합니다 (170페이지의 "AIR 응용 프로그램 서명" 참조).

AIR 설치 파일 만들기

❖ 명령 프롬프트에서 다음 명령을 한 줄에 입력합니다.

```
adt -package -storetype pkcs12 -keystore sampleCert.pfx HelloWorld.air
HelloWorld-app.xml HelloWorld.html AIRAliases.js
```

키 저장소 파일의 암호를 입력하라는 메시지가 나타납니다.

HelloWorld.air 인수는 ADT가 생성한 AIR 파일입니다. HelloWorld-app.xml은 응용 프로그램 설명자 파일입니다. 그 이후의 인수는 응용 프로그램에 사용되는 파일입니다. 이 예제에서는 파일 두 개만 사용하지만 원하는 모든 파일과 디렉토리를 포함할 수 있습니다. ADT에서 기본 내용 파일인 HelloWorld.html이 패키지에 포함되어 있는지 확인합니다. 하지만 AIRAliases.js를 포함하는 것을 잊어버리면 응용 프로그램이 작동하지 않습니다.

AIR 패키지가 만들어지면 해당 패키지 파일을 두 번 클릭하여 응용 프로그램을 설치하고 실행할 수 있습니다. 셸이나 명령 윈도우에서 명령으로 AIR 파일 이름을 입력할 수도 있습니다.

다음 단계

AIR에서 HTML 및 JavaScript 코드는 대개 일반적인 웹 브라우저에서와 동일하게 동작합니다. 실제로 AIR는 Safari 웹 브라우저에 사용되는 것과 동일한 WebKit 렌더링 엔진을 사용합니다. 하지만 AIR에서 HTML 응용 프로그램을 개발할 때 반드시 알아야 할 중요한 차이점이 있습니다. 이러한 차이점 및 기타 중요한 주제에 대한 자세한 내용은 Programming HTML and JavaScript를 참조하십시오.

Flex SDK를 사용하여 첫 번째 데스크톱 AIR 응용 프로그램 만들기

Adobe® AIR® 작동 방식을 직접 빠르게 알아보려면 다음에 나와 있는 지침에 따라 Flex SDK를 사용하여 간단한 SWF 기반 AIR "Hello World" 응용 프로그램을 만드십시오. 이 자습서에서는 Flex SDK에서 제공하는 명령줄 도구를 사용하여 AIR 응용 프로그램을 컴파일, 테스트 및 패키지화하는 방법을 보여 줍니다(Flex SDK에 AIR SDK가 포함되어 있음).

시작하려면 런타임을 설치하고 Adobe® Flex™를 설정해야 합니다. 이 자습서에서는 **AMXMLC** 컴파일러, ADL(**AIR Debug Launcher**) 및 ADT(**AIR Developer Tool**)를 사용합니다. 이러한 프로그램은 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다(17페이지의 “Flex SDK 설정” 참조).

AIR 응용 프로그램 설명자 파일 만들기

이 단원에서는 응용 프로그램 설명자를 만드는 방법을 설명합니다. 응용 프로그램 설명자는 다음과 같은 구조를 갖는 XML 파일입니다.

```
<application xmlns="...">
    <id>...</id>
    <versionNumber>...</versionNumber>
    <filename>…</filename>
    <initialWindow>
        <content>…</content>
        <visible>…</visible>
        <width>…</width>
        <height>…</height>
    </initialWindow>
</application>
```

1 HelloWorld-app.xml이라는 XML 파일을 만들어 프로젝트 디렉토리에 저장합니다.

2 AIR 네임스페이스 특성을 포함하여 <application> 요소를 추가합니다.

<application xmlns="http://ns.adobe.com/air/application/2.7"> 네임스페이스의 마지막 세그먼트인 "2.7"은 응용 프로그램에 필요한 런타임의 버전을 지정합니다.

3 <id> 요소를 추가합니다.

<id>samples.flex.HelloWorld</id> 응용 프로그램 ID는 제작자 ID(응용 프로그램 패키지에 서명하는 데 사용된 인증서에서 파생됨)와 함께 응용 프로그램을 고유하게 식별합니다. 권장되는 형식은 "com.company.AppName"과 같이 DNS 스타일과 반대 방향인, 점으로 구분된 문자열입니다. 응용 프로그램 ID는 설치, 개인 응용 프로그램 파일 시스템 저장소 디렉토리 액세스, 개인 암호화 저장소 액세스 및 응용 프로그램 간 통신에 사용됩니다.

4 <versionNumber> 요소를 추가합니다.

<versionNumber>1.0</versionNumber> 사용자에게 설치하는 응용 프로그램의 버전을 알려 줍니다.

참고: AIR 2 이하를 사용하는 경우 <versionNumber> 요소 대신 <version> 요소를 사용해야 합니다.

5 <filename> 요소를 추가합니다.

<filename>HelloWorld</filename> 응용 프로그램 실행 파일, 설치 디렉토리 및 운영 체제의 참조와 유사한 항목에 사용되는 이름입니다.

6 초기 응용 프로그램 윈도우의 속성을 지정하는 다음과 같은 자식 요소를 포함하는 <initialWindow> 요소를 추가합니다.

<content>HelloWorld.swf</content> AIR에서 로드할 루트 SWF 파일을 식별합니다.

<visible>true</visible> 윈도우를 즉시 보이게 만듭니다.

<width>400</width> 윈도우 폭을 픽셀 단위로 설정합니다.

<height>200</height> 윈도우 높이를 설정합니다.

7 파일을 저장합니다. 전체 응용 프로그램 설명자 파일은 다음과 같은 형태가 됩니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://ns.adobe.com/air/application/2.7">
    <id>samples.flex.HelloWorld</id>
    <versionNumber>0.1</versionNumber>
    <filename>HelloWorld</filename>
    <initialWindow>
        <content>HelloWorld.swf</content>
        <visible>true</visible>
        <width>400</width>
        <height>200</height>
    </initialWindow>
</application>
```

이 예제에서는 사용 가능한 응용 프로그램 속성 중 일부만 설정합니다. 윈도우 크롬, 윈도우 크기, 투명도, 기본 설치 디렉토리, 연결된 파일 유형, 응용 프로그램 아이콘 등을 지정할 수 있는 전체 응용 프로그램 속성을 보려면 183페이지의 “AIR 응용 프로그램 설명자 파일”을 참조하십시오.

응용 프로그램 코드 작성

참고: SWF 기반 AIR 응용 프로그램은 MXML 또는 Adobe® ActionScript® 3.0을 사용하여 정의된 기본 클래스를 사용할 수 있습니다. 아래 예제에서는 MXML 파일을 사용하여 기본 클래스를 정의합니다. 기본 ActionScript 클래스를 사용하여 AIR 응용 프로그램을 만드는 과정도 이와 비슷합니다. 단, MXML 파일을 SWF 파일로 컴파일하는 대신 ActionScript 클래스 파일을 컴파일합니다. ActionScript를 사용할 때는 기본 클래스가 flash.display.Sprite를 확장해야 합니다.

모든 Flex 기반 응용 프로그램과 마찬가지로, Flex 프레임워크로 구축된 AIR 응용 프로그램에는 기본 MXML 파일이 포함되어 있습니다. 데스크톱 AIR 응용 프로그램은 Application 구성 요소 대신 **WindowedApplication** 구성 요소를 루트 요소로 사용합니다. WindowedApplication 구성 요소는 응용 프로그램과 해당 초기 윈도우를 제어하기 위한 속성, 메서드 및 이벤트를 제공합니다. AIR가 여러 윈도우를 지원하지 않는 플랫폼 및 프로파일에서는 Application 구성 요소를 계속 사용하십시오. 모바일 Flex 응용 프로그램에서는 View 또는 TabbedViewNavigatorApplication 구성 요소를 사용할 수도 있습니다.

다음 절차에 따라 Hello World 응용 프로그램을 만듭니다.

1 텍스트 편집기를 사용하여 HelloWorld.mxml이라는 파일을 만들고 다음 MXML 코드를 추가합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                       xmlns:s="library://ns.adobe.com/flex/spark"
                       xmlns:mx="library://ns.adobe.com/flex/mx"
                       title="Hello World">
</s:WindowedApplication>
```

2 그런 다음 응용 프로그램에 Label 구성 요소를 추가합니다(WindowedApplication 태그 안에 배치).

3 Label 구성 요소의 text 속성을 "Hello AIR"로 설정합니다.

4 텍스트를 항상 가운데에 맞추도록 레이아웃 제한 사항을 설정합니다.

다음 예제는 지금까지 작성한 코드를 보여 줍니다.

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                       xmlns:s="library://ns.adobe.com/flex/spark"
                       xmlns:mx="library://ns.adobe.com/flex/mx"
                       title="Hello World">

    <s:Label text="Hello AIR" horizontalCenter="0" verticalCenter="0"/>
</s:WindowedApplication>
```

응용 프로그램 컴파일

응용 프로그램을 실행하고 디버깅하려면 먼저 amxmlc 컴파일러를 사용하여 MXML 코드를 SWF 파일로 컴파일하십시오. amxmlc 컴파일러는 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다. 필요한 경우 컴퓨터의 경로 환경을 Flex SDK bin 디렉토리를 포함하도록 설정할 수 있습니다. 경로를 설정하면 명령줄에서 유틸리티를 더 쉽게 실행할 수 있습니다.

1 명령 셸 또는 터미널을 열고 AIR 응용 프로그램의 프로젝트 폴더로 이동합니다.

2 다음 명령을 입력합니다.

```
amxmlc HelloWorld.mxml
```

amxmlc를 실행하면 HelloWorld.swf가 생성되며, 응용 프로그램의 컴파일된 코드가 이 파일에 포함됩니다.

참고: 응용 프로그램이 컴파일되지 않을 경우 구문 오류나 맞춤법 오류를 수정하십시오. amxmlc 컴파일러를 실행하는 데 사용되는 콘솔 윈도우에 오류 및 경고가 표시됩니다.

자세한 내용은 141페이지의 “AIR용 MXML 및 ActionScript 소스 파일 컴파일”을 참조하십시오.

응용 프로그램 테스트

명령줄에서 응용 프로그램을 실행하고 테스트하려면 ADL(AIR Debug Launcher)을 사용하여 해당 응용 프로그램 설명자 파일을 통해 응용 프로그램을 시작합니다. ADL은 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다.

❖ 명령 프롬프트에서 다음 명령을 입력합니다.

```
adl HelloWorld-app.xml
```

그러면 다음 그림과 비슷한 AIR 응용 프로그램이 표시됩니다.

Label 컨트롤의 horizontalCenter 및 verticalCenter 속성을 사용하여 텍스트를 윈도우 가운데에 배치합니다. 다른 모든 데스크톱 응용 프로그램과 마찬가지로 윈도우를 이동하거나 윈도우의 크기를 조정할 수 있습니다.

자세한 내용은 144페이지의 “ADL(AIR Debug Launcher)”을 참조하십시오.

AIR 설치 파일 만들기

응용 프로그램이 성공적으로 실행되면 ADT 유틸리티를 사용하여 응용 프로그램을 AIR 설치 파일로 패키지화할 수 있습니다. AIR 설치 파일은 사용자에게 응용 프로그램을 배포할 수 있도록 응용 프로그램의 모든 파일을 포함하는 보관 파일입니다. 패키지화된 AIR 파일을 설치하려면 먼저 Adobe AIR를 설치해야 합니다.

응용 프로그램 보안을 유지하기 위해 모든 AIR 설치 파일은 디지털 서명이 되어야 합니다. ADT나 다른 인증서 생성 도구에서는 개발 용도로 사용할 기본적인 자체 서명 인증서를 생성할 수 있습니다. 상업용 인증 기관에서 상업용 코드 서명 인증서를 구입할 수도 있습니다. 사용자가 자체 서명된 AIR 파일을 설치할 경우 설치 과정에서 제작자가 "알 수 없음"으로 표시됩니다. 이것은 자체 서명된 인증서가 AIR 파일이 만들어진 이후로 변경되지 않았다는 것만 보장하기 때문입니다. 악의적인 AIR 파일을 자체 서명하여 응용 프로그램으로 제공하는 것 자체를 막을 방법은 없습니다. 따라서, 공개적으로 배포되는 AIR 파일에는 검증 가능한 상용 인증서를 사용하는 것이 좋습니다. AIR 보안 문제에 대한 개요는 AIR 보안(ActionScript 개발자용) 또는 AIR security(HTML 개발자용)을 참조하십시오.

자체 서명 인증서 및 키 쌍 생성

❖ 명령 프롬프트에서 다음 명령을 입력합니다. ADT 실행 파일은 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다.

```
adt -certificate -cn SelfSigned 1024-RSA sampleCert.pfx samplePassword
```

이 예제에서는 인증서에 설정할 수 있는 최소한의 특성만 사용합니다. 키 유형은 **1024-RSA** 또는 **2048-RSA**여야 합니다 (170페이지의 “AIR 응용 프로그램 서명” 참조).

AIR 패키지 만들기

❖ 명령 프롬프트에서 다음 명령을 한 줄에 입력합니다.

```
adt -package -storetype pkcs12 -keystore sampleCert.pfx HelloWorld.air
HelloWorld-app.xml HelloWorld.swf
```

키 저장소 파일의 암호를 입력하라는 메시지가 나타납니다. 암호를 입력하고 Enter 키를 누릅니다. 보안상의 이유로 암호 문자는 표시되지 않습니다.

HelloWorld.air 인수는 ADT가 생성한 AIR 파일입니다. HelloWorld-app.xml은 응용 프로그램 설명자 파일입니다. 그 이후의 인수는 응용 프로그램에 사용되는 파일입니다. 이 예제에서는 파일 세 개만 사용하지만 원하는 수의 파일 및 디렉토리를 포함할 수 있습니다.

AIR 패키지가 만들어지면 해당 패키지 파일을 두 번 클릭하여 응용 프로그램을 설치하고 실행할 수 있습니다. 셸이나 명령 윈도우에서 명령으로 AIR 파일 이름을 입력할 수도 있습니다.

자세한 내용은 48페이지의 “데스크톱 AIR 설치 파일 패키지화”를 참조하십시오.

Flex SDK를 사용하여 첫 번째 Android용 AIR 응용 프로그램 만들기

시작하려면 AIR 및 Flex SDK가 설치 및 설정되어 있어야 합니다. 이 자습서에서는 Flex SDK 및 ADL(**AIR Debug Launcher**)의 **AMXMLC** 그리고 AIR SDK의 ADT(**AIR Developer Tool**)를 사용합니다. 17페이지의 “Flex SDK 설정”을 참조하십시오.

또한 Android 개발자: SDK 설치에 설명된 대로 Android 웹 사이트에서 Android SDK를 다운로드하여 설치해야 합니다.

참고: iPhone 개발에 대한 자세한 내용은 Flash Professional CS5를 사용하여 Hello World iPhone 응용 프로그램 만들기를 참조하십시오.

AIR 응용 프로그램 설명자 파일 만들기

이 단원에서는 응용 프로그램 설명자를 만드는 방법을 설명합니다. 응용 프로그램 설명자는 다음과 같은 구조를 갖는 XML 파일입니다.

```
<application xmlns="...">
    <id>...</id>
    <versionNumber>...</versionNumber>
    <filename>…</filename>
    <initialWindow>
        <content>…</content>
    </initialWindow>
    <supportedProfiles>...</supportedProfiles>
</application>
```

1 HelloWorld-app.xml이라는 XML 파일을 만들어 프로젝트 디렉토리에 저장합니다.

2 AIR 네임스페이스 특성을 포함하여 <application> 요소를 추가합니다.

<application xmlns="http://ns.adobe.com/air/application/2.7"> 네임스페이스의 마지막 세그먼트인 "2.7"은 응용 프로그램에 필요한 런타임의 버전을 지정합니다.

3 <id> 요소를 추가합니다.

<id>samples.android.HelloWorld</id> 응용 프로그램 ID는 제작자 ID(응용 프로그램 패키지에 서명하는 데 사용된 인증서에서 파생됨)와 함께 응용 프로그램을 고유하게 식별합니다. 권장되는 형식은 "com.company.AppName"과 같이 DNS 스타일과 반대 방향인, 점으로 구분된 문자열입니다.

4 <versionNumber> 요소를 추가합니다.

<versionNumber>0.0.1</versionNumber> 사용자에게 설치하는 응용 프로그램의 버전을 알려 줍니다.

5 <filename> 요소를 추가합니다.

<filename>HelloWorld</filename> 응용 프로그램 실행 파일, 설치 디렉토리 및 운영 체제의 참조와 유사한 항목에 사용되는 이름입니다.

6 초기 응용 프로그램 윈도우의 속성을 지정하는 다음과 같은 자식 요소를 포함하는 <initialWindow> 요소를 추가합니다.

<content>HelloWorld.swf</content> AIR에서 로드할 루트 HTML 파일을 식별합니다.

7 <supportedProfiles> 요소를 추가합니다.

<supportedProfiles>mobileDevice</supportedProfiles> 응용 프로그램이 모바일 프로파일에서만 실행된다는 것을 지정합니다.

8 파일을 저장합니다. 전체 응용 프로그램 설명자 파일은 다음과 같은 형태가 됩니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://ns.adobe.com/air/application/2.7">
    <id>samples.android.HelloWorld</id>
    <versionNumber>0.0.1</versionNumber>
    <filename>HelloWorld</filename>
    <initialWindow>
        <content>HelloWorld.swf</content>
    </initialWindow>
    <supportedProfiles>mobileDevice</supportedProfiles>
</application>
```

이 예제에서는 사용 가능한 응용 프로그램 속성 중 일부만 설정합니다. 응용 프로그램 설명자 파일에서 사용할 수 있는 다른 설정도 있습니다. 예를 들어 <fullScreen>true</fullScreen>를 initialWindow 요소에 추가하여 전체 화면 응용 프로그램을 만들 수 있습니다. Android에서 원격 디버깅 및 액세스 제어된 기능을 사용하려면 응용 프로그램 설명자에 Android 권한도 추가해야 합니다. 이 간단한 응용 프로그램에서는 권한이 필요하지 않기 때문에 지금은 추가할 필요가 없습니다.

자세한 내용은 64페이지의 "모바일 응용 프로그램 속성 설정"을 참조하십시오.

응용 프로그램 코드 작성

HelloWorld.as라는 파일을 만들고 텍스트 편집기를 사용하여 다음 코드를 추가합니다.

```
package
{
    import flash.display.Sprite;
    import flash.text.TextField;

    public class HelloWorld extends Sprite
    {
        public function HelloWorld()
        {
            var textField:TextField = new TextField();
            textField.text = "Hello, World!";
            stage.addChild( textField );
        }
    }
}
```

응용 프로그램 컴파일

응용 프로그램을 실행하고 디버깅하려면 먼저 amxmlc 컴파일러를 사용하여 MXML 코드를 SWF 파일로 컴파일하십시오. amxmlc 컴파일러는 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다. 필요한 경우 컴퓨터의 경로 환경을 Flex SDK bin 디렉토리를 포함하도록 설정할 수 있습니다. 경로를 설정하면 명령줄에서 유틸리티를 더 쉽게 실행할 수 있습니다.

1 명령 셸 또는 터미널을 열고 AIR 응용 프로그램의 프로젝트 폴더로 이동합니다.

2 다음 명령을 입력합니다.

```
amxmlc HelloWorld.as
```

amxmlc를 실행하면 HelloWorld.swf가 생성되며, 응용 프로그램의 컴파일된 코드가 이 파일에 포함됩니다.

참고: 응용 프로그램이 컴파일되지 않을 경우 구문 오류나 맞춤법 오류를 수정하십시오. amxmlc 컴파일러를 실행하는 데 사용되는 콘솔 윈도우에 오류 및 경고가 표시됩니다.

자세한 내용은 141페이지의 “AIR용 MXML 및 ActionScript 소스 파일 컴파일”을 참조하십시오.

응용 프로그램 테스트

명령줄에서 응용 프로그램을 실행하고 테스트하려면 ADL(AIR Debug Launcher)을 사용하여 해당 응용 프로그램 설명자 파일을 통해 응용 프로그램을 시작합니다. ADL은 AIR 및 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다.

❖ 명령 프롬프트에서 다음 명령을 입력합니다.

```
adl HelloWorld-app.xml
```

자세한 내용은 91페이지의 “ADL을 사용하는 장치 시뮬레이션”을 참조하십시오.

APK 패키지 파일 만들기

응용 프로그램이 성공적으로 실행되면 ADT 유틸리티를 사용하여 응용 프로그램을 APK 패키지 파일로 패키지화할 수 있습니다. APK 패키지 파일은 사용자에게 배포할 수 있는 기본 Android 응용 프로그램 파일 포맷입니다.

모든 Android 응용 프로그램에는 서명이 필요합니다. AIR 파일과 달리 관례적으로 자체 서명된 인증서를 사용하여 Android 응용 프로그램에 서명합니다. Android 운영 체제는 응용 프로그램 개발자의 ID를 설정하려고 시도하지 않습니다. ADT에서 생성된 인증서를 사용하여 Android 패키지에 서명할 수 있습니다. Android Market에 전송된 응용 프로그램에 사용된 인증서는 유효 기간이 적어도 25년이어야 합니다.

자체 서명 인증서 및 키 쌍 생성

❖ 명령 프롬프트에서 다음 명령을 입력합니다. ADT 실행 파일은 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다.

```
adt -certificate -validityPeriod 25 -cn SelfSigned 1024-RSA sampleCert.pfx samplePassword
```

이 예제에서는 인증서에 설정할 수 있는 최소한의 특성만 사용합니다. 키 유형은 **1024-RSA** 또는 **2048-RSA**여야 합니다 (157페이지의 “ADT certificate 명령” 참조).

AIR 패키지 만들기

❖ 명령 프롬프트에서 다음 명령을 한 줄에 입력합니다.

```
adt -package -target apk -storetype pkcs12 -keystore sampleCert.p12 HelloWorld.apk HelloWorld-app.xml
HelloWorld.swf
```

키 저장소 파일의 암호를 입력하라는 메시지가 나타납니다. 암호를 입력하고 Enter 키를 누릅니다.

자세한 내용은 85페이지의 “모바일 AIR 응용 프로그램 패키지화”를 참조하십시오.

AIR 런타임 설치

Android Market으로부터 최신 버전의 AIR 런타임을 장치에 설치할 수 있습니다. SDK에 포함된 런타임을 장치 또는 Android 에뮬레이터에 설치할 수도 있습니다.

❖ 명령 프롬프트에서 다음 명령을 한 줄에 입력합니다.

```
adt -installRuntime -platform android -platformsdk
```

-platformsdk 플래그를 Android SDK 디렉토리에 설정합니다(tools 폴더의 부모 지정).

ADT가 SDK에 들어 있는 Runtime.apk를 설치합니다.

자세한 내용은 98페이지의 “개발용 AIR 런타임 및 응용 프로그램 설치”를 참조하십시오.

AIR 응용 프로그램 설치

❖ 명령 프롬프트에서 다음 명령을 한 줄에 입력합니다.

```
adt -installApp -platform android -platformsdk path-to-android-sdk -package path-to-app
```

-platformsdk 플래그를 Android SDK 디렉토리에 설정합니다(tools 폴더의 부모 지정).

자세한 내용은 98페이지의 “개발용 AIR 런타임 및 응용 프로그램 설치”를 참조하십시오.

장치 또는 에뮬레이터의 스크린에서 응용 프로그램 아이콘을 눌러서 응용 프로그램을 실행할 수 있습니다.

6장: 데스크톱용 AIR 응용 프로그램 개발

데스크톱 AIR 응용 프로그램을 개발하기 위한 작업 과정

AIR 응용 프로그램을 개발하기 위한 기본 작업 과정은 대부분의 일반 개발 모델과 동일합니다. 코딩하고, 컴파일하고, 테스트하고 그리고 주기가 끝날 때는 설치 프로그램 파일로 패키지화합니다.

Flash, Flex 및 ActionScript를 사용하여 응용 프로그램 코드를 작성하고 Flash Professional, Flash Builder 또는 mxmlc 및 compc 명령줄 컴파일러를 사용하여 컴파일할 수 있습니다. HTML 및 JavaScript를 사용하여 응용 프로그램 코드를 작성하고 컴파일 단계를 건너뛸 수도 있습니다.

ADL 도구를 사용하여 데스크톱 AIR 응용 프로그램을 테스트할 수 있습니다. 이 도구는 먼저 패키지화 및 설치하지 않고도 응용 프로그램을 실행할 수 있습니다. Flash Professional, Flash Builder, Dreamweaver 및 Aptana IDE 모두가 Flash 디버거와 통합됩니다. 명령줄에서 ADL을 사용할 때 디버거 도구인 FDB를 수동으로 실행할 수도 있습니다. ADL 자체는 오류를 표시하고 명령문 출력을 추적합니다.

모든 AIR 응용 프로그램은 설치 파일로 패키지화되어야 합니다. 다음에 해당하지 않는 경우 플랫폼의 영향을 받지 않는 AIR 파일 형식을 사용하는 것이 좋습니다.

- NativeProcess 클래스 같은 플랫폼에 독립적인 API에 액세스해야 합니다.
- 응용 프로그램에 기본 확장이 사용됩니다.

그럴 경우에는 AIR 응용 프로그램을 플랫폼 고유의 기본 설치 프로그램 파일로 패키지화하면 됩니다.

SWF 기반 응용 프로그램

1. MXML 또는 ActionScript 코드를 작성합니다.
2. 아이콘 비트맵 파일 등 필요한 에셋을 만듭니다.
3. 응용 프로그램 설명자를 만듭니다.
4. ActionScript 코드를 컴파일합니다.
5. 응용 프로그램을 테스트합니다.
6. **air** 대상을 사용하여 AIR 파일로 패키지화 및 서명합니다.

HTML 기반 응용 프로그램

1. HTML 및 JavaScript 코드를 작성합니다.
2. 아이콘 비트맵 파일 등 필요한 에셋을 만듭니다.
3. 응용 프로그램 설명자를 만듭니다.
4. 응용 프로그램을 테스트합니다.
5. **air** 대상을 사용하여 AIR 파일로 패키지화 및 서명합니다.

AIR 응용 프로그램을 위한 기본 설치 프로그램 만들기

1. 코드를 작성합니다(ActionScript 또는 HTML 및 JavaScript).
2. 아이콘 비트맵 파일 등 필요한 에셋을 만듭니다.
3. **extendedDesktop** 프로파일을 지정하는 응용 프로그램을 설명자를 만듭니다.

4 ActionScript 코드를 컴파일합니다.

5 응용 프로그램을 테스트합니다.

6 **native** 대상을 사용하여 각 대상 플랫폼에서 응용 프로그램을 패키지화합니다.

참고: 대상 플랫폼용 기본 설치 프로그램은 해당 플랫폼에서 만들어야 합니다. 예를 들어 Windows 설치 프로그램을 Mac에서 만들 수는 없습니다. VMWare와 같은 가상 시스템을 사용하여 동일한 컴퓨터 하드웨어에서 여러 플랫폼을 실행할 수 있습니다.

전용 런타임 번들이 있는 AIR 응용 프로그램 만들기

1 코드를 작성합니다(ActionScript 또는 HTML 및 JavaScript).

2 아이콘 비트맵 파일 등 필요한 에셋을 만듭니다.

3 **extendedDesktop** 프로파일을 지정하는 응용 프로그램을 설명자를 만듭니다.

4 ActionScript 코드를 컴파일합니다.

5 응용 프로그램을 테스트합니다.

6 **bundle** 대상을 사용하여 각 대상 플랫폼에서 응용 프로그램을 패키지화합니다.

7 번들 파일을 사용하여 설치 프로그램을 만듭니다. AIR SDK는 이러한 설치 프로그램을 만들기 위한 도구를 제공하지 않지만 많은 타사 도구 키트를 사용할 수 있습니다.

참고: 대상 플랫폼용 번들은 해당 플랫폼에서 만들어야 합니다. 예를 들어 Windows 번들을 Mac에서 만들 수는 없습니다. VMWare와 같은 가상 시스템을 사용하여 동일한 컴퓨터 하드웨어에서 여러 플랫폼을 실행할 수 있습니다.

데스크톱 응용 프로그램 속성 설정

응용 프로그램 설명자 파일에서 기본 응용 프로그램 속성을 설정합니다. 이 섹션에서는 데스크톱 AIR 응용 프로그램과 관련된 속성을 다룹니다. 응용 프로그램 설명자 파일의 요소는 183페이지의 “AIR 응용 프로그램 설명자 파일”에서 자세히 설명합니다.

필요한 AIR 런타임 버전

응용 프로그램 설명자 파일의 네임스페이스를 사용하여 응용 프로그램에 필요한 AIR 런타임의 버전을 지정합니다.

응용 프로그램에서 사용할 수 있는 기능을 결정하는 데 가장 큰 영향을 미치는 것은 application 요소에 할당된 네임스페이스입니다. 예를 들어 응용 프로그램에서 AIR 1.5 네임스페이스를 사용하고 사용자에게 AIR 3.0가 설치되어 있는 경우 응용 프로그램은 AIR 1.5 비헤이비어를 인식하며, 이는 해당 비헤이비어가 AIR 3.0에서 변경된 경우에도 마찬가지입니다. 네임스페이스를 변경하고 업데이트를 제작할 경우에만 응용 프로그램이 새로운 비헤이비어 및 기능에 액세스하게 됩니다. 이 정책의 주요 예외는 보안 및 WebKit 변경 사항입니다.

루트 application 요소의 xmlns 특성을 사용하여 네임스페이스를 지정합니다.

```
<application xmlns="http://ns.adobe.com/air/application/3.0">
```

기타 도움말 항목

188페이지의 “application”

응용 프로그램 ID

제작하는 각 응용 프로그램에 대해 여러 설정이 고유해야 합니다. 고유 설정에는 ID, 이름 및 파일 이름이 포함됩니다.

```
<id>com.example.MyApplication</id>
<name>My Application</name>
<filename>MyApplication</filename>
```

기타 도움말 항목

203페이지의 “id”

198페이지의 “filename”

211페이지의 “name”

응용 프로그램 버전

AIR 2.5 미만의 AIR 버전에서는 version 요소에서 응용 프로그램을 지정합니다. 어떠한 문자열이라도 사용할 수 있습니다. AIR 런타임은 문자열을 해석하지 않습니다. 따라서 “2.0”은 “1.0”보다 높은 버전으로 취급되지 않습니다.

```
<!-- AIR 2 or earlier -->
<version>1.23 Beta 7</version>
```

AIR 2.5 이상에서는 versionNumber 요소에서 응용 프로그램 버전을 지정합니다. version 요소는 더 이상 사용할 수 없습니다. versionNumber 값을 지정할 때는 최대 세 개의 숫자로 이루어진 시퀀스를 사용해야 하며 이때 각 숫자는 점으로 분리해야 합니다(예: “0.1.2”). 버전 번호의 각 세그먼트는 최대 세 자리까지 가능합니다. 즉, “999.999.999”가 허용되는 가장 큰 버전 번호입니다. 번호에 세 세그먼트를 모두 포함해야 하는 것은 아닙니다. 따라서 “1” 및 “1.0”도 유효한 버전 번호입니다.

versionLabel 요소를 사용하여 버전에 대한 레이블을 지정할 수도 있습니다. 버전 레이블을 추가하면 AIR 응용 프로그램 설치 대화 상자 등의 위치에서 버전 번호 대신 표시됩니다.

```
<!-- AIR 2.5 and later -->
<versionNumber>1.23.7<versionNumber>
<versionLabel>1.23 Beta 7</versionLabel>
```

기타 도움말 항목

218페이지의 “version”

219페이지의 “versionLabel”

219페이지의 “versionNumber”

기본 윈도우 속성

AIR는 데스크톱에서 응용 프로그램을 시작할 때 윈도우를 만들고 기본 SWF 파일 또는 HTML 페이지를 그 안에 로드합니다. AIR는 initialWindow 요소의 자식 요소를 사용하여 이 초기 응용 프로그램 윈도우의 모양 및 비헤이비어를 제어합니다.

- **content** - initalWindow 요소의 content 자식에 있는 기본 응용 프로그램 SWF 파일입니다. 데스크톱 프로파일에 있는 장치를 대상으로 지정하는 경우 SWF 또는 HTML 파일을 사용할 수 있습니다.

  ```
  <initialWindow>
      <content>MyApplication.swf</content>
  </initialWindow>
  ```

  파일을 AIR 패키지에 포함해야 합니다(ADT 또는 IDE 사용). 단순히 응용 프로그램 설명자에 있는 이름을 참조하는 것만으로는 파일이 패키지에 자동으로 포함되지 않습니다.

- **depthAndStencil** - 심도 또는 스텐실 버퍼를 사용하도록 지정합니다. 일반적으로 이러한 버퍼는 3D 내용으로 작업하는 경우에 사용합니다.

  ```
  <depthAndStencil>true</depthAndStencil>
  ```

- **height** - 초기 윈도우의 높이입니다.

- **maximizable** - 윈도우를 최대화하기 위한 시스템 크롬이 표시되는지 여부입니다.
- **maxSize** - 허용되는 최대 크기입니다.
- **minimizable** - 윈도우를 최소화하기 위한 시스템 크롬이 표시되는지 여부입니다.
- **minSize** - 허용되는 최소 크기입니다.
- **renderMode** - AIR 3 이상에서 데스크톱 응용 프로그램에 대한 렌더링 모드를 **auto**, **cpu**, **direct** 또는 **gpu**로 설정할 수 있습니다. 이전 버전 AIR의 경우 이 설정은 데스크톱 플랫폼에서 무시됩니다. 런타임에는 renderMode 설정을 변경할 수 없습니다.
  - auto - 기본적으로 cpu 모드와 동일합니다.
  - cpu - 표시 객체가 소프트웨어의 표시 메모리에 렌더링 및 복사됩니다. 윈도우가 전체 화면 모드인 경우에만 StageVideo를 사용할 수 있습니다. Stage3D는 소프트웨어 렌더러를 사용합니다.
  - direct - 표시 객체는 런타임 소프트웨어에 의해 렌더링되지만 렌더링된 프레임을 표시 메모리에 복사(블록 전송)하는 것은 하드웨어 가속을 통해 수행됩니다. StageVideo를 사용할 수 있습니다. Stage3D는 가능한 경우 하드웨어 가속을 사용합니다. 윈도우 투명도가 true로 설정된 경우 윈도우는 소프트웨어 렌더링 및 블록 전송으로 변경됩니다.

    참고: 모바일 플랫폼용 AIR와 함께 Flash 내용의 GPU 가속 기능을 활용하려면 renderMode="gpu" 대신 renderMode="direct"(즉, Stage3D)를 사용하는 것이 좋습니다. Adobe에서는 Stage3D 기반 프레임워크인 Starling(2D)과 Away3D(3D)를 공식적으로 지원 및 권장합니다. Stage3D와 Starling/Away3D에 대한 자세한 내용은 http://gaming.adobe.com/getstarted/를 참조하십시오.
  - gpu - 가능한 경우 하드웨어 가속이 사용됩니다.
- **requestedDisplayResolution** - 고해상도 화면을 사용하는 MacBook Pro 컴퓨터에서 응용 프로그램이 표준 해상도를 사용할 것인지 고 해상도를 사용할 것인지를 지정합니다. 그 외 모든 플랫폼에서는 이 값이 무시됩니다. 값이 **standard**이면 화면의 각 스테이지 픽셀이 네 픽셀로 렌더링됩니다. 값이 **high**이면 화면의 각 스테이지 픽셀이 물리적인 한 픽셀에 해당합니다. 지정된 값은 모든 응용 프로그램 윈도우에 사용됩니다. 데스크톱 AIR 응용 프로그램에 requestedDisplayResolution 요소를 intialWindow 요소의 자식으로 사용하는 것은 AIR 3.6 이상에서 가능합니다.
- **resizable** - 윈도우 크기를 조정하기 위한 시스템 크롬이 표시되는지 여부입니다.
- **systemChrome** - 표준 운영 체제 윈도우 드레싱이 사용되는지 여부입니다. 런타임에는 윈도우의 systemChrome 설정을 변경할 수 없습니다.
- **title** - 윈도우의 제목입니다.
- **transparent** - 윈도우가 배경에 대해 알파-블렌드되는지 여부입니다. 투명도가 설정되어 있으면 윈도우가 시스템 크롬을 사용할 수 없습니다. 런타임에는 윈도우의 transparent 설정을 변경할 수 없습니다.
- **visible** - 윈도우가 생성되는 즉시 표시되는지 여부입니다. 기본적으로 처음에는 윈도우가 표시되지 않는데, 이를 통해 응용 프로그램에서는 윈도우를 표시하기 전에 먼저 내용을 그릴 수 있습니다.
- **width** - 윈도우의 폭입니다.
- **x** - 윈도우의 가로 위치입니다.
- **y** - 윈도우의 세로 위치입니다.

기타 도움말 항목

193페이지의 “content”

194페이지의 “depthAndStencil”

202페이지의 “height”

210페이지의 “maximizable”

210페이지의 “maxSize”

210페이지의 “minimizable”

210페이지의 “minimizable”

211페이지의 “minSize”

213페이지의 “renderMode”

214페이지의 “requestedDisplayResolution”

215페이지의 “resizable”

217페이지의 “systemChrome”

218페이지의 “title”

218페이지의 “transparent”

219페이지의 “visible”

220페이지의 “width”

220페이지의 “x”

221페이지의 “y”

데스크톱 기능

다음 요소는 데스크톱 설치 및 업데이트 기능을 제어합니다.

- customUpdateUI - 응용 프로그램을 업데이트하기 위한 대화 상자를 직접 만들어서 제공할 수 있도록 합니다. 기본값인 false로 설정되어 있으면 표준 AIR 대화 상자가 사용됩니다.
- fileTypes - 응용 프로그램에서 기본 시작 응용 프로그램으로 등록할 파일의 유형을 지정합니다. 다른 응용 프로그램이 이미 해당 파일 유형의 시작 응용 프로그램인 경우 AIR는 기존 등록을 무시하지 않습니다. 하지만 응용 프로그램은 NativeApplication 객체의 setAsDefaultApplication() 메서드를 사용하여 런타임에 등록을 무시할 수 있습니다. 기존 파일 유형 관계를 무시하기 전에 사용자의 허가를 구하는 것이 좋습니다.

  참고: 응용 프로그램을 전용 런타임 번들로 패키지화할 경우(-bundle 대상 사용) 파일 형식 등록이 무시됩니다. 지정된 파일 형식을 등록하려면 등록을 수행하는 설치 프로그램을 만들어야 합니다.
- installFolder - 응용 프로그램이 설치되는 표준 응용 프로그램 설치 폴더에 상대적인 경로를 지정합니다. 이 설정을 사용하여 사용자 정의 폴더 이름을 제공하고 여러 응용 프로그램을 공통 폴더 안에 그룹화할 수 있습니다.
- programMenuFolder - Windows [모든 프로그램] 메뉴의 메뉴 계층 구조를 지정합니다. 이 설정을 사용하여 여러 응용 프로그램을 공통 메뉴 안에 그룹화할 수 있습니다. 메뉴 폴더가 지정되어 있지 않으면 응용 프로그램 단축키가 기본 메뉴에 바로 추가됩니다.

기타 도움말 항목

194페이지의 “customUpdateUI”

200페이지의 “fileTypes”

207페이지의 “installFolder”

212페이지의 “programMenuFolder”

지원되는 프로파일

응용 프로그램이 데스크톱에만 적합한 경우 지원되는 프로파일 목록에서 해당 프로파일을 제외하는 방식으로 다른 프로파일에 있는 장치에 설치되지 않도록 할 수 있습니다. 응용 프로그램에서 NativeProcess 클래스 또는 기본 확장을 사용하는 경우 extendedDesktop 프로파일을 지원해야 합니다.

응용 프로그램 설명자에 supportedProfile 요소를 넣지 않으면 응용 프로그램이 정의된 모든 프로파일을 지원하는 것으로 가정합니다. 특정 프로파일 목록으로 응용 프로그램을 제한하려면 프로파일을 공백으로 분리하여 나열하십시오.

```
<supportedProfiles>desktop extendedDesktop</supportedProfiles>
```

desktop 및 extendedDesktop 프로파일에서 지원되는 ActionScript 클래스의 목록에 대해서는 223페이지의 “각 프로파일의 기능”을 참조하십시오.

기타 도움말 항목

216페이지의 “supportedProfiles”

필수 기본 확장

extendedDesktop 프로파일을 지원하는 응용 프로그램에서는 기본 확장을 사용할 수 있습니다.

응용 프로그램 설명자에서 AIR 응용 프로그램이 사용하는 모든 기본 확장을 선언합니다. 다음 예제에서는 두 개의 필수 기본 확장을 지정하기 위한 구문을 보여 줍니다.

```
<extensions>
    <extensionID>com.example.extendedFeature</extensionID>
    <extensionID>com.example.anotherFeature</extensionID>
</extensions>
```

extensionID 요소의 값은 확장 설명자 파일에서 id 요소의 값과 동일합니다. 확장 설명자 파일은 extension.xml이라는 XML 파일로, 이 파일은 기본 확장 개발자로부터 받는 ANE 파일에 패키지화되어 있습니다.

응용 프로그램 아이콘

데스크톱에서 응용 프로그램 설명자로 지정된 아이콘은 응용 프로그램 파일, 단축키 및 프로그램 메뉴 아이콘으로 사용됩니다. 응용 프로그램 아이콘은 16x16, 32x32, 48x48 및 128x128 픽셀 PNG 이미지 집합으로 제공되어야 합니다. 응용 프로그램 설명자 파일의 아이콘 요소에서 아이콘 파일의 경로를 지정합니다.

```
<icon>
    <image16x16>assets/icon16.png</image16x16>
    <image32x32>assets/icon32.png</image32x32>
    <image48x48>assets/icon48.png</image48x48>
    <image128x128>assets/icon128.png</image128x128>
</icon>
```

지정된 크기의 아이콘을 제공하지 않으면 그 다음으로 가장 큰 아이콘이 지정된 크기에 맞게 조정되어 사용됩니다. 아이콘을 하나도 제공하지 않으면 기본 시스템 아이콘이 사용됩니다.

기타 도움말 항목

203페이지의 “icon”

204페이지의 “imageNxN”

무시되는 설정

데스크톱의 응용 프로그램은 모바일 프로파일 기능에 적용되는 응용 프로그램 설정을 무시합니다. 무시되는 설정은 다음과 같습니다.

- android
- aspectRatio
- autoOrients
- fullScreen
- iPhone
- renderMode(AIR 3 이전)
- requestedDisplayResolution
- softKeyboardBehavior

데스크톱 AIR 응용 프로그램 디버깅

Flash Builder, Flash Professional, Dreamweaver 등의 IDE를 사용하여 응용 프로그램을 개발하는 경우에는 디버깅 도구가 일반적으로 내장되어 있습니다. 디버그 모드에서 실행함으로써 응용 프로그램을 간단히 디버깅할 수 있습니다. 디버깅을 직접 지원하는 IDE를 사용하고 있지 않은 경우에는 ADL(AIR Debug Launcher) 및 FDB(Flash Debugger)를 사용하여 응용 프로그램 디버깅을 지원할 수 있습니다.

기타 도움말 항목

De Monsters: 몬스터 디버거

256페이지의 “AIR HTML Introspector를 사용한 디버깅”

ADL을 사용하여 응용 프로그램 실행

ADL을 사용하여 패키지화하고 설치하지 않고도 AIR 응용 프로그램을 실행할 수 있습니다. 다음 예제에 나온 것처럼 응용 프로그램 설명자 파일을 ADL에 매개 변수로 전달합니다. 이를 위해 먼저 응용 프로그램에 있는 ActionScript 코드를 먼저 컴파일해야 합니다.

```
adl myApplication-app.xml
```

ADL은 추적 명령문, 런타임 예외 및 HTML 파싱 오류를 터미널 윈도우에 인쇄합니다. FDB 프로세스가 수신 연결을 대기 중일 경우 ADL은 디버거에 연결합니다.

또한 ADL을 사용하여 기본 확장을 사용하는 AIR 응용 프로그램을 디버깅할 수 있습니다. 예를 들면 다음과 같습니다.

```
adl -extdir extensionDirs myApplication-app.xml
```

기타 도움말 항목

144페이지의 “ADL(AIR Debug Launcher)”

trace 문 인쇄

trace 문을 ADL 실행에 사용되는 콘솔로 인쇄하려면 trace() 함수를 사용하여 코드에 trace 문을 추가합니다.

참고: trace() 문이 콘솔에 표시되지 않는 경우, mm.cfg 파일에 ErrorReportingEnable 또는 TraceOutputFileEnable을 지정하지 않았는지 확인하십시오. 이 파일의 플랫폼별 위치에 대한 자세한 내용은 mm.cfg 파일 편집을 참조하십시오.

ActionScript 예제:

```
//ActionScript
trace("debug message");
```

JavaScript 예제:

```
//JavaScript
air.trace("debug message");
```

JavaScript에서는 alert() 및 confirm() 함수를 사용하여 응용 프로그램에서 보내는 디버깅 메시지를 표시할 수 있습니다. 또한 catch되지 않은 JavaScript 예외와 구문 오류의 줄 번호가 콘솔로 인쇄됩니다.

참고: JavaScript 예제에 나오는 air 접두어를 사용하려면 AIRAliases.js 파일을 페이지로 가져와야 합니다. 이 파일은 AIR SDK의 frameworks 디렉토리에 있습니다.

Flash Debugger(FDB)에 연결

Flash Debugger를 사용하여 AIR 응용 프로그램을 디버깅하려면 FDB 세션을 시작한 다음 ADL을 사용하여 응용 프로그램을 시작합니다.

참고: SWF 기반 AIR 응용 프로그램에서는 ActionScript 소스 파일을 -debug 플래그로 컴파일해야 합니다. Flash Professional에서는 [제작 설정] 대화 상자에서 [디버깅 허용] 옵션을 선택합니다.

1. FDB를 시작합니다. FDB 프로그램은 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다.

   콘솔은 FDB 프롬프트인 <fdb>를 표시합니다.

2. run 명령 <fdb>run [Enter]를 실행합니다.
3. 다른 명령 또는 셸 콘솔에서 해당 응용 프로그램의 디버그 버전을 시작합니다.

   ```
   adl myApp.xml
   ```

4. FDB 명령을 사용하여 원하는 중단점을 설정합니다.
5. continue [Enter]를 입력합니다.

AIR 응용 프로그램이 SWF 기반인 경우 디버거는 ActionScript 코드의 실행만 제어합니다. AIR 응용 프로그램이 HTML 기반인 경우에는 디버거가 JavaScript 코드의 실행만 제어합니다.

디버거에 연결하지 않고 ADL을 실행하려면 -nodebug 옵션을 포함합니다.

```
adl myApp.xml -nodebug
```

FDB 명령에 대한 기본 정보를 보려면 help 명령을 실행합니다.

```
<fdb>help [Enter]
```

FDB 명령에 대한 자세한 내용은 Flex 설명서의 명령줄 디버거 명령 사용을 참조하십시오.

데스크톱 AIR 설치 파일 패키지화

모든 AIR 응용 프로그램은 최소한 응용 프로그램 설명자 파일 및 기본 SWF 또는 HTML 파일이 있어야 합니다. 응용 프로그램에 설치할 다른 에셋들도 AIR 파일로 패키지화해야 합니다.

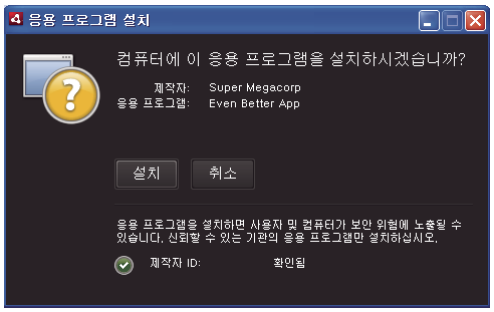
이 문서에서는 SDK에 포함된 명령줄 도구를 사용하여 AIR 응용 프로그램을 패키지화하는 내용에 대해 설명합니다. Adobe 제작 도구 중 하나를 사용하여 응용 프로그램을 패키지화하는 방법에 대한 자세한 내용은 다음을 참조하십시오.

- Adobe® Flex® Builder™: Packaging AIR applications with Flex Builder 참조
- Adobe® Flash® Builder™: Packaging AIR applications with Flash Builder 참조

- Adobe® Flash® Professional: Adobe AIR용으로 제작 참조
- Adobe® Dreamweaver®: Dreamweaver에서 AIR 응용 프로그램 만들기 참조

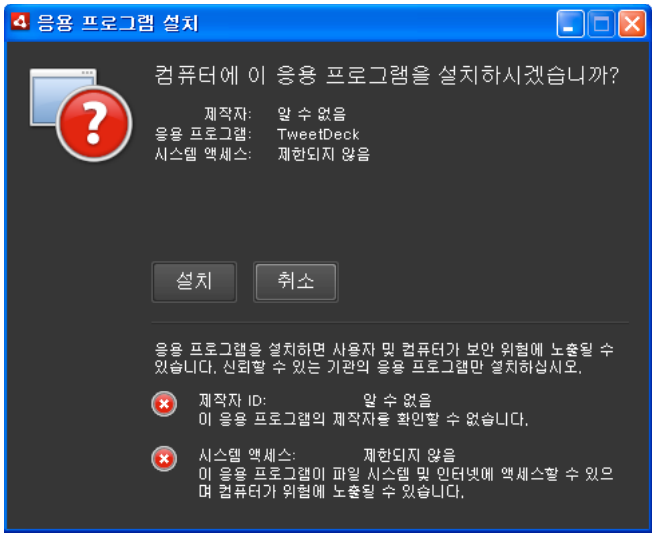
모든 AIR 설치 프로그램 파일은 디지털 인증서를 사용하여 서명해야 합니다. AIR 설치 프로그램은 서명을 사용하여 사용자가 응용 프로그램 파일에 서명한 이후 해당 파일이 변경되지 않았는지 확인합니다. 인증 기관의 코드 서명 인증서를 사용하거나 자체 서명된 인증서를 사용할 수 있습니다.

신뢰할 수 있는 인증 기관에서 발행된 인증서를 사용할 경우에는 응용 프로그램 사용자에게 제작자로서 자신의 ID에 대한 약간의 보증을 제공할 수 있습니다. 설치 대화 상자에는 자신의 ID가 인증 기관에 의해 확인되었다는 메시지가 표시됩니다.



신뢰할 수 있는 인증서로 서명된 응용 프로그램의 설치 확인 대화 상자

자체 서명된 인증서를 사용할 경우에는 사용자가 자신의 ID를 서명자로서 확인할 수 없습니다. 자체 서명된 인증서는 또한 패키지가 수정되지 않았음에 대한 보증도 약합니다. 사용자가 응용 프로그램을 받아 보기 전에 올바른 설치 파일이 위조된 파일로 대체될 수 있기 때문입니다. 설치 대화 상자에는 제작자의 ID를 확인할 수 없다는 메시지가 표시됩니다. 사용자는 응용 프로그램을 설치할 때 큰 보안 위험을 감수합니다.



자체 서명 인증서로 서명된 응용 프로그램의 설치 확인 대화 상자

ADT -package 명령을 사용하여 한 단계로 AIR 파일을 패키지 및 서명할 수 있습니다. -prepare 명령을 사용하여 서명되지 않은 중간 패키지를 만들고 별도의 단계에서 -sign 명령을 사용하여 중간 패키지에 서명할 수 있습니다.

참고: Java 버전 1.5 이상의 경우 PKCS12 인증서 파일을 보호하는 데 사용되는 암호에 상위 ASCII 문자를 사용할 수 없습니다. 코드 서명 인증서 파일을 만들거나 내보낼 경우에는 암호로 일반 ASCII 문자만 사용하십시오.

설치 패키지에 서명할 때 ADT는 자동으로 타임스탬프 기관 서버에 연결하여 시간을 확인합니다. 타임스탬프 정보는 AIR 파일에 포함됩니다. 확인된 타임스탬프가 들어 있는 AIR 파일은 나중에 언제든지 설치할 수 있습니다. ADT가 타임스탬프 서버에 연결할 수 없을 경우 패키지가 취소됩니다. 타임스탬프 옵션은 재정의가 가능하지만 타임스탬프가 없으면 설치 파일 서명에 사용된 인증서가 만료된 후에 AIR 응용 프로그램을 더 이상 설치할 수 없습니다.

기존 AIR 응용 프로그램을 업데이트하는 패키지를 만드는 중인 경우 원래 응용 프로그램과 동일한 인증서로 패키지를 서명해야 합니다. 원래 인증서가 갱신되었거나, 만료된 후 180일이 지나지 않았거나, 새 인증서로 바꾸려는 경우 마이그레이션 서명을 적용할 수 있습니다. 마이그레이션 서명은 응용 프로그램 AIR 파일을 새 인증서 및 이전 인증서 모두로 서명합니다. 156페이지의 "ADT migrate 명령"에 설명된 대로 -migrate 명령을 사용하여 마이그레이션 서명을 적용합니다.

중요: 원래 인증서가 만료된 후 마이그레이션 서명을 적용할 수 있는 유예 기간은 정확히 180일입니다. 마이그레이션 서명을 사용하지 않으면 기존 사용자가 새 버전을 설치하기 전에 기존 응용 프로그램을 제거해야 합니다. 유예 기간은 응용 프로그램 설명자 네임스페이스에서 AIR 버전 1.5.3 이상이 지정된 응용 프로그램에만 적용됩니다. 이전 AIR 런타임 버전을 대상으로 할 경우에는 유예 기간이 없습니다.

AIR 1.1 이전에는 마이그레이션 서명이 지원되지 않습니다. 마이그레이션 서명을 적용하려면 SDK 버전 1.1 이상으로 응용 프로그램을 패키지화해야 합니다.

AIR 파일을 사용하여 배포된 응용 프로그램은 데스크톱 프로파일 응용 프로그램으로 알려져 있습니다. 응용 프로그램 설명자 파일이 데스크톱 프로파일을 지원하지 않을 경우 ADT를 사용하여 AIR 응용 프로그램에 대한 기본 설치 프로그램을 패키지화할 수 없습니다. 응용 프로그램 설명자 파일에서 supportedProfiles 요소를 사용하여 프로파일을 제한할 수 있습니다. 222페이지의 "장치 프로파일" 및 216페이지의 "supportedProfiles"를 참조하십시오.

참고: 응용 프로그램 설명자 파일의 설정은 AIR 응용 프로그램의 ID 및 기본 설치 경로를 결정합니다. 183페이지의 "AIR 응용 프로그램 설명자 파일"을 참조하십시오.

제작자 ID

AIR 1.5.3부터는 제작자 ID가 더 이상 사용되지 않습니다. 원래 AIR 1.5.3 이상 버전에서 제작된 새 응용 프로그램에는 제작자 ID가 필요하지 않으며 이를 지정하지 않아야 합니다.

이전 버전의 AIR를 사용하여 제작된 응용 프로그램을 업데이트할 경우에는 응용 프로그램 설명자 파일에 원래 제작자 ID를 지정해야 합니다. 그렇지 않으면 응용 프로그램의 설치된 버전과 업데이트 버전이 다른 응용 프로그램으로 취급됩니다. 다른 ID를 사용하거나 publisherID 태그를 생략할 경우 사용자가 새 버전을 설치하기 전에 이전 버전을 제거해야 합니다.

원래 제작자 ID를 확인하려면 원래 응용 프로그램이 설치된 META-INF/AIR 하위 디렉토리에서 publisherid 파일을 찾아보십시오. 이 파일 내에 들어 있는 문자열이 제작자 ID입니다. 제작자 ID를 수동으로 지정하려면 응용 프로그램 설명자가 응용 프로그램 설명자 파일의 네임스페이스 선언에서 AIR 1.5.3 런타임 이상을 지정해야 합니다.

AIR 1.5.3 이전에 제작되었거나 AIR 1.5.3 SDK로 제작되었지만 응용 프로그램 설명자 네임스페이스에서 이전 AIR 버전이 지정된 응용 프로그램의 경우 제작자 ID가 서명 인증서를 기준으로 계산됩니다. 이 ID는 응용 프로그램 ID와 함께 응용 프로그램의 ID를 식별하는 데 사용됩니다. 제작자 ID(있을 경우)는 다음과 같은 용도로 사용됩니다.

- AIR 파일이 설치할 새 응용 프로그램보다 최신 파일인지 확인
- 암호화된 로컬 저장소를 위한 암호화 키의 일부
- 응용 프로그램 저장소 디렉토리 경로의 일부
- 로컬 연결을 위한 연결 문자열의 일부
- AIR 인 브라우저(in-browser) API를 통해 응용 프로그램을 호출하는 데 사용되는 ID 문자열의 일부
- OSID의 일부(사용자 정의 설치/제거 프로그램을 만들 때 사용됨)

AIR 1.5.3 이전에는 신규 또는 갱신된 인증서를 사용하는 마이그레이션 서명으로 응용 프로그램 업데이트를 서명할 경우 응용 프로그램의 제작자 ID가 변경될 수 있었습니다. 제작자 ID가 변경되면 해당 ID를 사용하는 모든 AIR 기능의 비헤이비어도 변경됩니다. 예를 들어 기존 암호화된 로컬 저장소에 포함된 데이터에 더 이상 액세스할 수 없으며, 응용 프로그램에 대한 로컬 연결을 만드는 모든 Flash 또는 AIR 인스턴스는 연결 문자열에 새로운 ID를 사용해야 합니다.

AIR 1.5.3 또는 그 이상에서는 제작자 ID가 서명 인증서를 기반으로 하지 않으며 응용 프로그램 설명자에 publisherID 태그가 포함된 경우에만 지정됩니다. 업데이트 AIR 패키지에 대해 지정된 제작자 ID가 현재 제작자 ID와 일치하지 않을 경우 응용 프로그램을 업데이트할 수 없습니다.

ADT를 사용하여 패키지화

AIR ADT 명령줄 도구를 사용하여 AIR 응용 프로그램을 패키지화할 수 있습니다. 패키지화하기 전에 모든 ActionScript, MXML 및 확장 코드를 컴파일해야 합니다. 코드 서명 인증서도 있어야 합니다.

ADT 명령 및 옵션에 대한 자세한 참조는 149페이지의 "ADT(AIR Developer Tool)"를 참조하십시오.

AIR 패키지 만들기

AIR 패키지를 만들려면 릴리스 빌드에 대해 대상 유형을 **air**로 설정하는 ADT package 명령을 사용하십시오.

```
adt -package -target air -storetype pkcs12 -keystore ../codesign.p12 myApp.air myApp-app.xml myApp.swf icons
```

이 예제에서는 ADT 도구의 경로가 명령줄 셸의 경로 정의에 있다고 가정합니다. 도움말은 277페이지의 "path 환경 변수"를 참조하십시오.

응용 프로그램 파일이 들어 있는 디렉토리에서 명령을 실행해야 합니다. 이 예제의 응용 프로그램 파일은 myApp-app.xml(응용 프로그램 설명자 파일), myApp.swf 및 icons 디렉토리입니다.

표시된 것과 같은 명령을 실행하면 ADT에서 키 저장소 암호를 묻습니다. 입력하는 암호 문자가 항상 표시되는 것은 아닙니다. 암호 문자를 표시하려면 입력을 마치고 Enter 키를 누르면 됩니다.

AIRI 파일로부터 AIR 패키지 만들기

AIRI 파일을 만들고 서명하여 설치 가능한 AIR 패키지를 만들 수 있습니다.

```
adt -sign -storetype pkcs12 -keystore ../codesign.p12 myApp.airi myApp.air
```

데스크톱 기본 설치 프로그램 패키지화

AIR 2에서는 ADT를 사용하여 AIR 응용 프로그램 배포에 대한 기본 응용 프로그램 설치 프로그램을 만들 수 있습니다. 예를 들어, Windows에서 AIR 응용 프로그램을 배포하기 위한 EXE 설치 프로그램 파일을 만들 수 있습니다. Mac OS에서는 AIR 응용 프로그램의 배포를 위한 DMG 설치 프로그램 파일을 만들 수 있습니다. AIR 2.5 및 AIR 2.6의 경우 Linux에서는 AIR 응용 프로그램의 배포를 위한 DEB 또는 RPM 설치 프로그램 파일을 만들 수 있습니다.

기본 응용 프로그램 설치 프로그램으로 설치된 응용 프로그램은 확장 데스크톱 프로파일 응용 프로그램이라고 합니다. 응용 프로그램 설명자 파일이 데스크톱 확장 프로파일을 지원하지 않을 경우 ADT를 사용하여 AIR 응용 프로그램에 대한 기본 설치 프로그램을 패키지화할 수 없습니다. 응용 프로그램 설명자 파일에서 supportedProfiles 요소를 사용하여 프로파일을 제한할 수 있습니다. 222페이지의 "장치 프로파일" 및 216페이지의 "supportedProfiles"를 참조하십시오.

AIR 응용 프로그램의 기본 설치 프로그램 버전은 다음 두 가지 방식으로 만들 수 있습니다.

- 응용 프로그램 설명자 파일 및 기타 소스 파일을 기반으로 기본 설치 프로그램을 만들 수 있습니다. 기타 소스 파일에는 SWF 파일, HTML 파일 및 기타 에셋이 포함될 수 있습니다.
- AIR 파일 기반 또는 AIRI 파일 기반으로 기본 설치 프로그램을 만들 수 있습니다.

생성하려는 기본 설치 프로그램 파일과 동일한 운영 체제에서 ADT를 사용해야 합니다. 따라서 Windows용 EXE 파일을 만들려면 Windows에서 ADT를 실행하십시오. Mac OS용 DMG 파일을 만들려면 Mac OS에서 ADT를 실행하십시오. Linux용 DEB 또는 RPG 파일을 만들려면 Linux의 AIR 2.6 SDK에서 ADT를 실행하십시오.

AIR 응용 프로그램을 배포하기 위해 기본 설치 프로그램을 만들 경우 이러한 응용 프로그램에는 다음과 같은 기능이 포함됩니다.

- NativeProcess 클래스를 이용하여 기본 프로세스를 실행하고 상호 작용할 수 있습니다. 자세한 내용은 다음 중 하나를 참조하십시오.
  - AIR의 기본 프로세스와 통신(ActionScript 개발자용)
  - Communicating with native processes in AIR(HTML 개발자용)
- 기본 확장을 사용할 수 있습니다.
- 이 클래스는 파일 유형에 관계없이 File.openWithDefaultApplication() 메서드를 사용하여 모든 파일을 열 수 있습니다. 파일은 해당 파일을 열도록 정의된 기본 시스템 응용 프로그램에서 열립니다. 기본 설치 프로그램으로 설치되지 **않은** 응용 프로그램에서는 사용할 수 없습니다. 자세한 내용은 언어 참조 설명서에서 File.openWithDefaultApplication() 항목에 대한 항목을 참조하십시오.

하지만 기본 설치 프로그램으로 패키지화된 응용 프로그램은 AIR 파일 포맷의 몇 가지 이점을 잃게 됩니다. 더 이상 모든 데스크톱 컴퓨터에 단일 파일을 배포할 수 없습니다. 내장 업데이트 기능(그리고 업데이터 프레임워크)이 작동하지 않습니다.

사용자가 기본 설치 프로그램 파일을 두 번 클릭하면 AIR 응용 프로그램이 설치됩니다. 필요한 Adobe AIR 버전이 시스템에 아직 설치되지 않은 경우 설치 프로그램이 네트워크에서 Adobe AIR 버전을 다운로드하여 먼저 설치합니다. 올바른 Adobe AIR 버전(필요한 경우)을 가져올 수 있는 네트워크 연결이 없는 경우 설치가 실패합니다. 또한 운영 체제가 Adobe AIR 2에서 지원되지 않으면 설치가 실패합니다.

참고: 설치된 응용 프로그램에서 파일이 실행 가능하도록 하려면 해당 응용 프로그램을 패키지화할 때 파일 시스템에서 해당 파일을 실행할 수 있어야 합니다. Mac 및 Linux에서는 필요한 경우 chmod를 사용하여 실행 가능 플래그를 설정할 수 있습니다.

응용 프로그램 소스 파일에서 기본 설치 프로그램 만들기

응용 프로그램에 대한 소스 파일에서 기본 설치 프로그램을 만들려면 단일 명령줄에서 다음 구문으로 -package 명령을 사용하십시오.

```
adt -package AIR_SIGNING_OPTIONS
    -target native
    [WINDOWS_INSTALLER_SIGNING_OPTIONS]
    installer_file
    app_xml
    [file_or_dir | -C dir file_or_dir | -e file dir ...] ...
```

이 구문은 AIR 파일을 패키지화하는 구문과 비슷합니다(기본 설치 프로그램 사용 안 함) 하지만 다음과 같은 몇 가지 차이점이 있습니다.

- -target native 옵션을 명령에 추가합니다. -target air를 지정한 경우 ADT는 기본 설치 프로그램 파일 대신 AIR 파일을 생성합니다.
- 대상 DMG 또는 EXE 파일을 installer_file로 지정합니다.
- 선택적으로 Windows에서는 구문 목록에 [WINDOWS_INSTALLER_SIGNING_OPTIONS]로 표시되는 두 번째 서명 옵션 집합을 추가할 수 있습니다. Windows에서는 AIR 파일 서명 외에도 Windows 설치 프로그램 파일을 서명할 수 있습니다. AIR 파일 서명에서와 동일한 인증서 유형 및 서명 옵션 구문을 사용합니다(161페이지의 "ADT 코드 서명 옵션" 참조). AIR 파일 및 설치 프로그램 파일을 서명하는 데 동일한 인증서를 사용하거나 다른 인증서를 지정할 수 있습니다. 사용자가 웹에서 서명된 Windows 설치 프로그램 파일을 다운로드한 경우 Windows는 인증서를 기반으로 파일 소스를 식별합니다.

-target 옵션 이외의 ADT 옵션에 대한 자세한 내용은 149페이지의 "ADT(AIR Developer Tool)"를 참조하십시오.

다음 예에서는 DMG 파일(Mac OS의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -package
    -storetype pkcs12
    -keystore myCert.pfx
    -target native
    myApp.dmg
    application.xml
    index.html resources
```

다음 예에서는 EXE 파일(Windows의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -package
    -storetype pkcs12
    -keystore myCert.pfx
    -target native
    myApp.exe
    application.xml
    index.html resources
```

다음 예에서는 EXE 파일을 만들고 서명합니다.

```
adt -package
    -storetype pkcs12
    -keystore myCert.pfx
    -target native
    -storetype pkcs12
    -keystore myCert.pfx
    myApp.exe
    application.xml
    index.html resources
```

기본 확장을 사용하는 응용 프로그램용 기본 설치 프로그램 만들기

응용 프로그램의 소스 파일과 응용 프로그램에 필요한 기본 확장 패키지에서 기본 설치 프로그램을 작성할 수 있습니다. -package 명령을 다음과 같은 구문으로 사용합니다(단일 명령줄에서).

```
adt -package AIR_SIGNING_OPTIONS
    -migrate MIGRATION_SIGNING_OPTIONS
    -target native
    [WINDOWS_INSTALLER_SIGNING_OPTIONS]
    installer_file
    app_xml
    -extdir extension-directory
    [file_or_dir | -C dir file_or_dir | -e file dir ...] ...
```

이 구문은 추가된 두 옵션을 제외하고 기본 설치 프로그램을 패키지화하는 데 사용된 구문과 동일합니다. -extdir ***extension-directory*** 옵션을 사용하여 응용 프로그램에 사용되는 ANE 파일(기본 확장)이 포함된 디렉토리를 지정합니다. 옵션 -migrate 플래그 및 MIGRATION_SIGNING_OPTIONS 매개 변수는 기본 코드 서명 인증서가 이전 버전에서 사용하는 인증서와 다를 경우 마이그레이션 서명으로 응용 프로그램의 업데이트에 서명할 때 사용합니다. 자세한 내용은 179페이지의 “업데이트된 버전의 AIR 응용 프로그램에 서명”을 참조하십시오.

ADT 옵션에 대한 자세한 내용은 149페이지의 “ADT(AIR Developer Tool)”를 참조하십시오.

다음 예제에서는 기본 확장을 사용하는 응용 프로그램에 대한 DMG 파일(Mac OS X용 기본 설치 프로그램 파일)을 만듭니다.

```
adt -package
    -storetype pkcs12
    -keystore myCert.pfx
    -target native
    myApp.dmg
    application.xml
    -extdir extensionsDir
    index.html resources
```

AIR 파일 또는 AIRI 파일에서 기본 설치 프로그램 만들기

ADT를 사용하여 AIR 파일 또는 AIRI 파일을 기반으로 기본 설치 프로그램 파일을 생성할 수 있습니다. AIR 파일을 기반으로 기본 설치 프로그램을 만들려면 단일 명령줄에 다음 구문으로 ADT -package 명령을 사용하십시오.

```
adt -package
    -target native
    [WINDOWS_INSTALLER_SIGNING_OPTIONS]
    installer_file
    air_file
```

이 구문은 AIR 응용 프로그램에 대한 소스 파일을 기반으로 기본 설치 프로그램을 만들 때의 구문과 비슷합니다. 그러나 다음과 같은 몇 가지 차이점이 있습니다.

- 응용 프로그램 설명자 파일 및 AIR 응용 프로그램에 대한 기타 소스 파일 대신 AIR 파일을 소스로 지정합니다.
- AIR 파일은 이미 서명되어 있으므로 AIR 파일에 대해 서명 옵션을 지정하지 않습니다.

AIRI 파일을 기반으로 기본 설치 프로그램을 만들려면 단일 명령줄에 다음 구문으로 ADT -package 명령을 사용하십시오.

```
adt AIR_SIGNING_OPTIONS
    -package
    -target native
    [WINDOWS_INSTALLER_SIGNING_OPTIONS]
    installer_file
    airi_file
```

이 구문은 AIR 파일을 기반으로 기본 설치 프로그램을 만들 때의 구문과 비슷합니다. 하지만 다음과 같은 몇 가지 차이점이 있습니다.

- AIRI 파일을 소스로 지정합니다.
- 대상 AIR 응용 프로그램에 대한 서명 옵션을 지정합니다.

다음 예에서는 AIR 파일을 기반으로 DMG 파일(Mac OS의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -package -target native myApp.dmg myApp.air
```

다음 예에서는 AIR 파일을 기반으로 EXE 파일(Windows의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -package -target native myApp.exe myApp.air
```

다음 예에서는 EXE 파일(AIR 파일 기반)을 만들고 서명합니다.

```
adt -package -target native -storetype pkcs12 -keystore myCert.pfx myApp.exe myApp.air
```

다음 예에서는 AIRI 파일을 기반으로 DMG 파일(Mac OS의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -storetype pkcs12 -keystore myCert.pfx -package -target native myApp.dmg myApp.airi
```

다음 예에서는 AIRI 파일을 기반으로 EXE 파일(Windows의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -storetype pkcs12 -keystore myCert.pfx -package -target native myApp.exe myApp.airi
```

다음 예제에서는 AIRI 파일을 바탕으로 EXE 파일을 만들고 AIR 및 기본 Windows 서명을 모두 사용하여 서명합니다.

```
adt -package -storetype pkcs12 -keystore myCert.pfx -target native -storetype pkcs12 -keystore myCert.pfx
myApp.exe myApp.airi
```

데스크톱 컴퓨터용 전용 런타임 번들 패키지화

전용 런타임 번들은 전용 버전의 런타임과 함께 응용 프로그램 코드를 포함하는 패키지입니다. 이 방식으로 패키지화된 응용 프로그램은 사용자 컴퓨터의 다른 곳에 설치된 공유 런타임 대신에 번들된 런타임을 사용합니다.

생성된 번들은 Windows의 응용 프로그램 파일과 Mac OS의 .app 번들이 포함된 자체 포함형 폴더입니다. 대상 운영 체제에서 실행되는 동안 해당 운영 체제에 맞는 번들을 생성해야 합니다. VMWare와 같은 가상 시스템을 사용하여 한 대의 컴퓨터에서 여러 운영 체제를 실행할 수 있습니다.

응용 프로그램을 설치하지 않고 해당 폴더 또는 번들에서 실행할 수 있습니다.

장점

- 자체 포함형 응용 프로그램이 생성됩니다.
- 설치를 위해 인터넷에 액세스할 필요가 없습니다.
- 응용 프로그램이 런타임 업데이트와 분리됩니다.
- 기업에서 특정 응용 프로그램 및 런타임 조합을 인증할 수 있습니다.
- 기존의 소프트웨어 배포 모델을 지원합니다.
- 별도의 런타임 재배포가 필요하지 않습니다.
- NativeProcess API를 사용할 수 있습니다.
- 기본 확장을 사용할 수 있습니다.
- File.openWithDefaultApplication() 함수를 제한 없이 사용할 수 있습니다.
- 설치할 필요 없이 USB 또는 광 디스크에서 실행할 수 있습니다.

단점

- Adobe에서 보안 패치를 게시하는 경우 중요한 보안 픽스를 사용자가 자동으로 사용할 수 없습니다.
- .air 파일 포맷을 사용할 수 없습니다.
- 필요한 경우 고유한 설치 프로그램을 만들어야 합니다.
- AIR 업데이트 API 및 프레임워크가 지원되지 않습니다.
- 웹 페이지에서 AIR 응용 프로그램을 설치 및 시작하기 위한 AIR 인 브라우저(in-browser) API가 지원되지 않습니다.
- Windows의 경우 설치 프로그램에서 파일 등록을 처리해야 합니다.
- 응용 프로그램 디스크가 더 많이 사용됩니다.

Windows에서 전용 런타임 번들 만들기

Windows 전용 런타임 번들을 만들려면 Windows 운영 체제에서 실행되는 동안 응용 프로그램을 패키지화해야 합니다. ADT **bundle** 대상을 사용하여 응용 프로그램을 패키지화합니다.

```
adt -package
    -keystore ..\cert.p12 -storetype pkcs12
    -target bundle
    myApp
    myApp-app.xml
    myApp.swf icons resources
```

이 명령은 myApp이라는 디렉토리에 번들을 만듭니다. 이 디렉토리에는 응용 프로그램에 대한 파일과 런타임 파일이 포함됩니다. 프로그램을 이 폴더에서 직접 실행할 수 있습니다. 그러나 프로그램 메뉴 항목, 레지스터 파일 형식 또는 URI 스킴 처리기를 만들려면 필요한 레지스트리 항목을 설정하는 설치 프로그램을 만들어야 합니다. AIR SDK에는 이러한 설치 프로그램을 만들기 위한 도구가 없지만 상용 또는 무료 오픈 소스 설치 프로그램 도구 키트를 비롯한 여러 타사 옵션을 사용할 수 있습니다.

명령줄에서 -target bundle 항목 뒤에 두 번째 서명 옵션 집합을 지정하여 WIndows에서 기본 실행 파일에 서명할 수 있습니다. 이러한 서명 옵션은 기본 Windows 서명을 적용할 때 사용할 개인 키 및 연결된 인증서를 식별합니다. 일반적으로 AIR 코드 서명 인증서를 사용할 수 있습니다. 기본 실행 파일만 서명되며, 응용 프로그램과 함께 패키지화된 추가 실행 파일은 이 프로세스에서 서명되지 않습니다.

파일 형식 연결

Windows에서 응용 프로그램을 공용 또는 사용자 정의 파일 형식과 연결하려면 설치 프로그램에서 적절한 레지스트리 항목을 설정해야 합니다. 파일 형식은 응용 프로그램 설명자 파일의 fileTypes 요소에도 나열되어야 합니다.

Windows 파일 형식에 대한 자세한 내용은 MSDN 라이브러리: 파일 형식 및 파일 연결을 참조하십시오.

URI 처리기 등록

응용 프로그램에서 지정된 URI 스킴을 사용하여 URL의 시작을 처리하려면 설치 프로그램에서 필요한 레지스트리 항목을 설정해야 합니다.

URI 스킴을 처리하기 위해 응용 프로그램을 등록하는 방법에 대한 자세한 내용은 MSDN 라이브러리: 응용 프로그램을 URL 프로토콜에 등록을 참조하십시오.

Mac OS X에서 전용 런타임 번들 만들기

Mac OS X용 전용 런타임 번들을 만들려면 Macintosh 운영 체제에서 실행되는 동안 응용 프로그램을 패키지화해야 합니다. ADT **bundle** 대상을 사용하여 응용 프로그램을 패키지화합니다.

```
adt -package
    -keystore ../cert.p12 -storetype pkcs12
    -target bundle
    myApp.app
    myApp-app.xml
    myApp.swf icons resources
```

이 명령은 myApp.app라는 응용 프로그램 번들을 만듭니다. 이 번들에는 응용 프로그램에 대한 파일과 런타임 파일이 포함됩니다. myApp.app 아이콘을 두 번 클릭하여 응용 프로그램을 실행하고 Applications 폴더 같은 적절한 위치로 끌어 설치할 수 있습니다. 그러나 파일 형식 또는 URI 스킴 처리기를 등록하려면 응용 프로그램 패키지 내에서 속성 목록 파일을 편집해야 합니다.

배포를 위해 디스크 이미지 파일(.dmg)을 만들 수 있습니다. Adobe AIR SDK에서는 전용 런타임 번들용 dmg 파일을 만들기 위한 도구를 제공하지 않습니다.

파일 형식 연결

Mac OS X에서 응용 프로그램을 공용 또는 사용자 정의 파일 형식과 연결하려면 번들에서 info.plist 파일을 편집하여 CFBundleDocumentTypes 속성을 설정해야 합니다. Mac OS X 개발자 라이브러리: 정보 속성 목록 키 참조, CFBundleURLTypes를 참조하십시오.

URI 처리기 등록

응용 프로그램에서 지정된 URI 스킴을 사용하여 URL의 시작을 처리하려면 번들에서 info.plist 파일을 편집하여 CFBundleURLTypes 속성을 설정해야 합니다. Mac OS X 개발자 라이브러리: 정보 속성 목록 키 참조, CFBundleDocumentTypes를 참조하십시오.

데스크톱 컴퓨터를 위한 AIR 패키지 배포

AIR 응용 프로그램은 응용 프로그램 코드 및 모든 에셋을 포함하는 AIR 패키지로 배포될 수 있습니다. 다운로드, 전자 메일 또는 물리적 미디어(예: CD-ROM)와 같은 일반적인 수단을 통해 이 패키지를 배포할 수 있습니다. 사용자는 AIR 파일을 두 번 클릭하여 응용 프로그램을 설치할 수 있습니다. AIR 인 브라우저(in-browser) API(웹 기반 ActionScript 라이브러리)를 사용하여 사용자가 웹 페이지에서 하나의 링크를 통해 AIR 응용 프로그램(그리고 필요한 경우 Adobe® AIR®)을 설치하도록 할 수 있습니다.

AIR 응용 프로그램은 기본 설치 프로그램(즉, Windows의 EXE 파일, Mac의 DMG 파일, Linux의 DEB 또는 RPM 파일)으로 패키지화되고 배포될 수 있습니다. 기본 설치 패키지는 관련 플랫폼 규약에 따라 배포되고 설치될 수 있습니다. 응용 프로그램을 기본 패키지로 배포하면 AIR 파일 포맷의 몇 가지 이점을 잃게 됩니다. 구체적으로, 더 이상 대부분의 플랫폼에서 단일 설치 파일을 사용할 수 없고, AIR 업데이트 프레임워크를 사용할 수 없으며, 인 브라우저 API를 사용할 수 없습니다.

데스크톱에서 AIR 응용 프로그램 설치 및 실행

AIR 파일은 단순하게 수신자에게 보내면 됩니다. 예를 들어 AIR 파일을 전자 메일 첨부 파일이나 웹 페이지의 링크로 보낼 수 있습니다.

사용자는 AIR 응용 프로그램을 다운로드한 후 다음 지침을 따라 설치합니다.

1 AIR 파일을 두 번 클릭합니다.

Adobe AIR가 컴퓨터에 이미 설치되어 있어야 합니다.

2 [설치] 윈도우에서 기본 설정을 선택한 상태로 두고 [계속]을 클릭합니다.

Windows에서 AIR는 자동으로 다음을 수행합니다.

- 응용 프로그램을 Program Files 디렉토리에 설치
- 응용 프로그램에 대한 바탕 화면 단축키 만들기
- [시작] 메뉴 단축키 만들기
- [제어판]의 [프로그램 추가/제거]에서 응용 프로그램에 대한 항목 추가

Mac OS에서 기본적으로 응용 프로그램은 Applications 디렉토리에 추가됩니다.

응용 프로그램이 이미 설치된 경우 설치 프로그램은 응용 프로그램의 기존 버전을 열 것인지, 아니면 다운로드한 AIR 파일의 버전으로 업데이트할 것인지를 사용자가 선택할 수 있게 합니다. 설치 프로그램은 AIR 파일의 응용 프로그램 ID 및 제작자 ID를 사용하여 응용 프로그램을 식별합니다.

3 설치가 완료되면 [완료]를 클릭합니다.

Mac OS에서는 업데이트된 버전의 응용 프로그램을 설치하려면 사용자에게 응용 프로그램 디렉토리에 설치할 수 있는 적절한 시스템 권한이 필요합니다. Windows 및 Linux에서는 사용자에게 관리 권한이 필요합니다.

응용 프로그램은 ActionScript 또는 JavaScript를 통해 새 버전을 설치할 수도 있습니다. 자세한 내용은 234페이지의 “AIR 응용 프로그램 업데이트”를 참조하십시오.

AIR 응용 프로그램이 설치되면 사용자는 다른 데스크톱 응용 프로그램과 마찬가지로 단순히 응용 프로그램 아이콘을 두 번 클릭하여 응용 프로그램을 실행합니다.

- Windows에서는 바탕 화면이나 폴더에 설치된 응용 프로그램 아이콘을 두 번 클릭하거나 [시작] 메뉴에서 응용 프로그램을 선택합니다.
- Linux에서는 바탕 화면이나 폴더에 설치된 응용 프로그램 아이콘을 두 번 클릭하거나 응용 프로그램 메뉴에서 응용 프로그램을 선택합니다.
- Mac OS에서는 응용 프로그램이 설치된 폴더에서 해당 응용 프로그램을 두 번 클릭합니다. 기본 설치 디렉토리는 /Applications 디렉토리입니다.

참고: AIR 2.6 이하용으로 개발된 AIR 응용 프로그램만 Linux에 설치할 수 있습니다.

AIR 연속 설치 기능을 사용하면 사용자는 웹 페이지의 링크를 클릭하여 AIR 응용 프로그램을 설치할 수 있습니다. AIR 브라우저 호출 기능을 사용하면 사용자는 웹 페이지의 링크를 클릭하여 설치된 AIR 응용 프로그램을 실행할 수 있습니다. 이러한 기능에 대해서는 다음 단원에서 설명합니다.

웹 페이지에서 데스크톱 AIR 응용 프로그램 설치 및 실행

AIR 인 브라우저 API를 사용하면 웹 페이지에서 AIR 응용 프로그램을 설치하고 실행할 수 있습니다. AIR 인 브라우저 API는 Adobe에서 호스팅하는 SWF 라이브러리인 **air.swf**에서 제공됩니다. AIR SDK에는 이 라이브러리를 사용하여 AIR 응용 프로그램(그리고 필요한 경우 런타임)을 설치, 업데이트 또는 실행하는 샘플 "배지" 응용 프로그램이 포함되어 있습니다. 제공된 샘플 배지를 수정할 수도 있고, 온라인 air.swf 라이브러리를 바로 사용하는 배지 웹 응용 프로그램을 직접 만들 수도 있습니다.

모든 AIR 응용 프로그램은 웹 페이지 배지를 통해 설치될 수 있습니다. 하지만 응용 프로그램 설명자 파일에 <allowBrowserInvocation>true</allowBrowserInvocation> 요소를 포함하는 응용 프로그램만 웹 배지를 통해 실행될 수 있습니다.

기타 도움말 항목

226페이지의 "AIR.SWF 인 브라우저 API"

데스크톱 컴퓨터에서의 엔터프라이즈 배포

IT 관리자는 Adobe AIR 런타임 및 AIR 응용 프로그램을 표준 데스크톱 배포 도구를 사용하여 자동으로 설치할 수 있습니다. IT 관리자는 다음을 수행할 수 있습니다.

- Microsoft SMS, IBM Tivoli 또는 부트스트래퍼 사용 자동 설치를 허용하는 기타 배포 도구와 같은 도구를 사용하여 Adobe AIR 런타임 자동 설치
- 런타임을 배포하는 데 사용된 것과 같은 도구를 사용하여 AIR 응용 프로그램 자동 설치

자세한 내용은 **Adobe AIR 관리자 안내서(http://www.adobe.com/go/learn_air_admin_guide_kr)**를 참조하십시오.

데스크톱 컴퓨터의 설치 로그

설치 로그는 AIR 런타임 자체 또는 AIR 응용 프로그램이 설치될 때 기록됩니다. 로그 파일을 검사하면 발생하는 모든 설치 또는 업데이트 문제의 원인을 확인하는 데 도움이 됩니다.

로그 파일은 다음 위치에 생성됩니다.

- Mac: 표준 시스템 로그(/private/var/log/system.log)

  Console 응용 프로그램(일반적으로 Utilities 폴더에 있음)을 열어서 Mac 시스템을 볼 수 있습니다.
- Windows XP: C:\Documents and Settings\<사용자 이름>\Local Settings\Application Data\Adobe\AIR\logs\Install.log
- Windows Vista, Windows 7: C:\Users\<사용자 이름>\AppData\Local\Adobe\AIR\logs\Install.log
- Linux: /home/<사용자 이름>/.appdata/Adobe/AIR/Logs/Install.log

참고: 이러한 로그 파일은 AIR 2 미만의 AIR 버전에서는 생성되지 않습니다.

7장: 휴대 장치용 AIR 응용 프로그램 개발

휴대 장치의 AIR 응용 프로그램은 기본 응용 프로그램으로 배포됩니다. 그리고 AIR 파일 포맷이 아니라 장치의 응용 프로그램 포맷을 사용합니다. 현재 AIR는 Android APK 패키지 및 iOS IPA 패키지를 지원합니다. 응용 프로그램 패키지의 릴리스 버전을 만든 후에 표준 플랫폼 메커니즘을 통해 응용 프로그램을 배포할 수 있습니다. Android의 경우 이는 일반적으로 Android Market을 의미하며, iOS의 경우에는 Apple App Store를 의미합니다.

AIR SDK 그리고 Flash Professional, Flash Builder 또는 다른 ActionScript 개발 도구를 사용하여 휴대 장치용 AIR 응용 프로그램을 만들 수 있습니다. HTML 기반 모바일 AIR 응용 프로그램은 현재 지원되지 않습니다.

참고: RIM(Research In Motion) BlackBerry Playbook에서는 AIR 개발에 사용할 수 있는 고유한 SDK를 제공합니다. Playbook 개발에 대한 자세한 내용은 RIM: BlackBerry Tablet OS 개발을 참조하십시오.

참고: 이 문서에서는 AIR 2.6 이상 SDK를 사용하여 iOS 응용 프로그램을 개발하는 방법에 대해 설명합니다. AIR 2.6 이상으로 만든 응용 프로그램은 iOS 4 이상을 실행하는 iPhone 3G, iPhone 4 및 iPad 장치에 설치할 수 있습니다. 이전 버전의 iOS를 위한 AIR 응용 프로그램을 개발하려면 iPhone 응용 프로그램 만들기에 설명된 대로 AIR 2 Packager for iPhone을 사용해야 합니다.

유용한 개인 정보 보호 방법에 대한 자세한 내용은 Adobe AIR SDK 개인 정보 보호 가이드를 참조하십시오.

AIR 응용 프로그램을 실행하기 위한 전체 시스템 요구 사항을 확인하려면 Adobe AIR 시스템 요구 사항을 참조하십시오.

개발 환경 설정

모바일 플랫폼에는 일반 AIR, Flex 및 Flash 개발 환경 설정 외에도 추가적인 설정 요구 사항이 있습니다. 기본 AIR 개발 환경 설정에 대한 자세한 내용은 15페이지의 “AIR 개발용 Adobe Flash Platform 도구”를 참조하십시오.

Android 설정

AIR 2.6 이상에서는 일반적으로 Android를 위한 특수한 설정이 필요 없습니다. AIR SDK(lib/android/bin 폴더)에는 Android ADB 도구가 포함되어 있습니다. AIR SDK는 ADB 도구를 사용하여 장치에서 응용 프로그램 패키지를 설치, 제거 및 실행합니다. ADB를 사용하여 시스템 로그를 볼 수도 있습니다. Android 에뮬레이터를 만들고 실행하려면 별도의 Android SDK를 다운로드해야 합니다.

응용 프로그램에서 응용 프로그램 설명자의 <manifestAdditions> 요소에 현재 버전의 AIR에서 유효한 것으로 인식하지 않는 요소를 추가하는 경우에는 보다 최신 버전의 Android SDK를 설치해야 합니다. AIR_ANDROID_SDK_HOME 환경 변수 또는 -platformsdk 명령줄 인수를 SDK의 파일 경로로 설정합니다. AIR 패키지 도구인 ADT에서는 이 SDK를 사용하여 <manifestAdditions> 요소의 항목에 대한 유효성을 검사합니다.

AIR 2.5에서는 Google에서 Android SDK의 별도 복사본을 다운로드해야 합니다. Android SDK 폴더를 참조하도록 AIR_ANDROID_SDK_HOME 환경 변수를 설정할 수 있습니다. 이 환경 변수를 설정하지 않으면 ADT 명령줄의 -platformsdk 인수에서 Android SDK의 경로를 지정해야 합니다.

기타 도움말 항목

169페이지의 “ADT 환경 변수”

277페이지의 “path 환경 변수”

iOS 설정

장치에서 iOS 응용 프로그램을 설치 및 테스트하고 해당 응용 프로그램을 배포하려면 Apple iOS Developer 프로그램(유료 서비스)에 가입해야 합니다. iOS Developer 프로그램에 가입하면 iOS Provisioning Portal에 액세스하여 테스트 및 후속 배포를 위해 응용 프로그램을 장치에 설치하는 데 필요한 항목 및 파일을 얻을 수 있습니다. Apple에서 제공하는 이들 항목 및 파일은 다음과 같습니다.

- 개발 및 배포 인증서
- 응용 프로그램 ID
- 개발 및 배포 프로비저닝 파일

모바일 응용 프로그램 설계 고려 사항

휴대 장치는 사용 면에서 그리고 물리적 특성으로 인해 신중한 코딩과 디자인이 요구됩니다. 예를 들어 최대한 빨리 실행되도록 코드를 간소화하는 것이 매우 중요합니다. 물론 이를 바탕으로 코드 최적화도 이루어져야 합니다. 장치 제한 내에서 작동하는 지능적인 디자인도 시각적 프레젠테이션이 렌더링 시스템에 과부하를 일으키지 않도록 하는 데 도움이 될 수 있습니다.

코드

코드를 보다 빨리 실행되도록 만드는 것이 항상 유리하지만, 대부분의 휴대 장치는 비교적 프로세서 속도가 느리기 때문에 코드를 간략하게 만드는 것이 더 이득일 수도 있습니다. 또한 휴대 장치는 거의 항상 배터리 전원으로 가동됩니다. 보다 적은 작업으로 같은 결과를 얻을 수 있다면 배터리 전원도 덜 사용하게 됩니다.

디자인

응용 프로그램의 사용자 환경을 디자인할 때는 작은 스크린 크기, 터치스크린 상호 작용 모드 그리고 끊임없이 바뀌는 모바일 사용자 환경 등의 요인을 고려해야 합니다.

코드와 디자인을 동시에

응용 프로그램에서 애니메이션을 사용하는 경우에는 렌더링 최적화가 매우 중요합니다. 하지만 대개의 경우 코드 최적화만으로는 부족합니다. 응용 프로그램의 시각적인 측면을 코드가 효율적으로 렌더링할 수 있도록 디자인해야 합니다.

중요한 최적화 기술은 Flash Platform의 성능 최적화 가이드에 설명되어 있습니다. 이 가이드에서 다루는 기술은 모든 Flash 및 AIR 내용에 적용되지만 휴대 장치에서 실행되는 응용 프로그램을 개발할 때 필수적입니다.

- Paul Trani: 모바일용 Flash 개발의 유용한 팁과 기법
- roguish: 모바일용 AIR의 GPU 테스트 응용 프로그램
- Jonathan Campos: AIR for Android 응용 프로그램의 최적화 기법
- Charles Schulze: AIR 2.6 게임 개발: iOS 포함

응용 프로그램 수명 주기

응용 프로그램의 포커스가 다른 응용 프로그램으로 넘어갈 때 AIR는 프레임 속도를 초당 4프레임으로 떨어뜨리고 그래픽 렌더링을 중지합니다. 이 프레임 속도 아래에서는 스트리밍 네트워크와 소켓 연결이 끊어지는 경향이 있습니다. 응용 프로그램에서 이러한 연결을 사용하지 않는 경우에는 프레임 속도를 더 낮게 조정할 수 있습니다.

적절할 때는 오디오 재생을 중지하고 Geolocation 및 Accelerometer 센서에 대한 리스너를 제거해야 합니다. AIR NativeApplication 객체는 활성화 및 비활성화 이벤트를 전달합니다. 이러한 이벤트를 사용하여 활성 및 백그라운드 상태 간의 전환을 관리하십시오.

대부분의 모바일 운영 체제는 경고 없이 백그라운드 응용 프로그램을 종료합니다. 응용 프로그램 상태를 자주 저장함으로써 응용 프로그램이 백그라운드에서 활성 상태로 돌아가거나 새로 실행될 때 합당한 상태로 복원될 수 있어야 합니다.

정보 밀도

휴대 장치 화면은 데스크톱보다 픽셀 밀도(인치당 픽셀 수)는 높고 물리적 크기는 더 작습니다. 같은 글꼴 크기를 사용해도 휴대 장치 스크린의 글자는 데스크톱 컴퓨터의 글자보다 작습니다. 가독성을 높이려면 더 큰 글꼴을 사용해야 하는 경우가 많습니다. 일반적으로 최소한 글꼴 크기가 14포인트 이상이어야 쉽게 읽을 수 있습니다.

휴대 장치는 이동 중이나 조명 상태가 나쁜 곳에서도 자주 사용됩니다. 따라서 현실적으로 가독성을 확보하면서 화면에 표시할 수 있는 정보의 양을 고려해야 합니다. 픽셀 크기가 같은 데스크톱 화면에 표시할 수 있는 정보의 양보다 적을 수밖에 없습니다.

또한 사용자가 스크린을 터치할 때 자신의 손가락과 손에 의해 디스플레이의 일부가 가려진다는 점도 고려하십시오. 사용자가 계속 사용해야 하는 대화형 요소는 스크린의 측면과 아래쪽에 배치하십시오.

텍스트 입력

많은 장치에서 가상 키보드를 사용하여 텍스트를 입력합니다. 가상 키보드는 화면의 일부분을 가리며 사용하기 번거로울 때가 많습니다. 키보드 이벤트(소프트 키 제외)에 의존하지 않도록 하십시오.

입력 테스트 필드를 사용하는 대체 방법을 구현해보십시오. 예를 들어 사용자가 숫자 값을 입력하도록 할 때는 텍스트 필드를 사용하지 않아도 됩니다. 대신 값을 높이거나 낮추는 버튼 2개만 제공하면 됩니다.

소프트 키

휴대 장치에는 다양한 수의 소프트 키가 포함되어 있습니다. 소프트 키는 다양한 기능을 수행하도록 프로그래밍할 수 있는 버튼입니다. 응용 프로그램에서 소프트 키에 대한 플랫폼 규약을 따르십시오.

화면 방향 변경

모바일 내용은 세로 또는 가로 방향으로 볼 수 있습니다. 따라서 응용 프로그램에서 화면 방향 변화를 어떻게 처리할지 고려해야 합니다. 자세한 내용은 스테이지 방향을 참조하십시오.

화면 보호

AIR는 비디오를 재생하는 동안 화면 보호 기능이 켜지는 것을 자동으로 차단하지 않습니다. AIR NativeApplication 객체의 systemIdleMode 속성을 사용하여 장치가 절전 모드로 들어가는지 여부를 제어할 수 있습니다. 일부 플랫폼의 경우 이 기능을 사용하려면 적절한 권한을 요청해야 합니다.

수신 전화 통화

AIR 런타임은 사용자가 전화를 걸거나 받을 때 오디오를 자동으로 음소거합니다. Android에서는 응용 프로그램이 백그라운드 상태일 때 오디오를 재생하는 경우 응용 프로그램 설명자에서 Android READ_PHONE_STATE 권한을 설정해야 합니다. 이렇게 하지 않으면 Android에서 런타임이 전화 통화를 감지하여 오디오를 자동으로 음소거하는 작동이 차단됩니다. 70페이지의 “Android 권한”을 참조하십시오.

히트 대상

사용자가 누르는 버튼과 기타 사용자 인터페이스 요소를 디자인할 때는 히트 대상의 크기를 고려하십시오. 이러한 요소는 터치 스크린에서 손가락으로 쉽게 활성화할 수 있을 만큼 충분히 커야 합니다. 또한 대상 간의 공간도 충분해야 합니다. 히트 대상 영역은 dpi가 높은 일반적인 휴대폰 스크린의 경우 각 사이드에서 약 44~57픽셀 사이여야 합니다.

응용 프로그램 패키지 설치 크기

휴대 장치는 일반적으로 데스크톱 컴퓨터보다 응용 프로그램 및 데이터를 설치할 수 있는 저장 공간이 훨씬 적습니다. 사용되지 않는 에셋 및 라이브러리를 제거하여 패키지 크기를 최소화하십시오.

Android에서는 응용 프로그램이 설치될 때 응용 프로그램 패키지가 개별 파일로 압축 해제되지 않습니다. 대신 에셋에 액세스하면 해당 에셋이 임시 저장소에 압축 해제됩니다. 이 압축 해제되는 에셋 저장소 공간을 최소화하려면 에셋이 완전히 로드되었을 때 파일 및 URL 스트림을 닫으십시오.

파일 시스템 액세스

모바일 운영 체제마다 각기 다른 파일 시스템 제한이 있으며 이러한 제한은 데스크톱 운영 체제에서 적용되는 제한과는 다른 경향이 있습니다. 따라서 파일 및 데이터를 저장하기에 알맞은 위치는 플랫폼마다 다릅니다.

파일 시스템의 차이로 인해 AIR File 클래스를 통해 제공되는 공통 디렉토리에 대한 바로 가기를 사용할 수 없는 경우도 있습니다. 다음 표에서는 Android 및 iOS에서 사용할 수 있는 바로 가기를 보여 줍니다.

	Android	**iOS**
File.applicationDirectory	URL을 통한 읽기 전용(기본 경로 아님)	읽기 전용
File.applicationStorageDirectory	사용 가능합니다.	사용 가능합니다.
File.cacheDirectory	사용 가능합니다.	사용 가능합니다.
File.desktopDirectory	sdcard의 루트	사용할 수 없습니다.
File.documentsDirectory	sdcard의 루트	사용 가능합니다.
File.userDirectory	sdcard의 루트	사용할 수 없습니다.
File.createTempDirectory()	사용 가능합니다.	사용 가능합니다.
File.createTempFile()	사용 가능합니다.	사용 가능합니다.

iOS 응용 프로그램에 관한 Apple 지침에는 상황에 따른 파일 저장 위치에 관한 구체적인 규칙이 명시되어 있습니다. 예를 들어 사용자 입력 데이터 또는 다시 생성하거나 다시 다운로드할 수 없는 데이터가 들어 있는 파일만 원격 백업용으로 지정된 디렉토리에 저장할 수 있다는 지침이 있습니다. 파일 백업 및 캐싱에 관한 Apple 지침을 준수하는 방법에 관한 자세한 내용은 파일 백업 및 캐싱 제어를 참조하십시오.

UI 구성 요소

Adobe에서는 모바일용으로 최적화된 버전의 Flex 프레임워크를 개발하였습니다. 자세한 내용은 Flex와 Flash Builder를 활용한 모바일 응용 프로그램 개발을 참조하십시오.

모바일 응용 프로그램에 적합한 커뮤니티 구성 요소 프로젝트도 사용할 수 있습니다. 그 중 일부는 다음과 같습니다.

- Josh Tynjala의 Starling용 Feathers UI 컨트롤
- Derrick Grigg의 skinnable version of Minimal Comps
- Todd Anderson의 as3flobile components

스테이지 3D 가속 그래픽 렌더링

AIR 3.2부터 모바일용 AIR은 스테이지 3D 가속 그래픽 렌더링을 지원합니다. Stage3D ActionScript API는 고급 2D 및 3D 기능을 구현하는 하위 수준 GPU 가속 API 집합입니다. 개발자는 이러한 하위 수준 API를 통해 보다 자유롭게 GPU 하드웨어 가속을 활용함으로써 성능을 대폭 끌어올릴 수 있습니다. 또한 Stage3D ActionScript API를 지원하는 게임 엔진을 사용할 수도 있습니다.

자세한 내용은 게임 엔진, 3D 및 스테이지 3D를 참조하십시오.

비디오 다듬기

성능 향상을 위해 AIR에서는 비디오 다듬기가 사용되지 않습니다.

기본 기능

AIR 3.0 이상

대부분의 모바일 플랫폼은 표준 AIR API를 통해 아직 액세스할 수 없는 기능을 제공합니다. AIR 3부터는 고유한 기본 코드 라이브러리를 사용하여 AIR을 확장할 수 있습니다. 이러한 기본 확장 라이브러리는 운영 체제 또는 특정 장치에서 사용할 수 있는 기능에 액세스할 수 있습니다. 기본 확장은 iOS에서 C로 작성되거나 Android에서 Java 또는 C로 작성될 수 있습니다. 기본 확장 개발에 대한 자세한 내용은 Adobe AIR용 기본 확장 소개를 참조하십시오.

휴대 장치용 AIR 응용 프로그램을 개발하기 위한 작업 과정

모바일(또는 기타) 장치용 AIR 응용 프로그램을 만들기 위한 작업 과정은 일반적으로 데스크톱 응용 프로그램을 만들기 위한 작업 과정과 매우 비슷합니다. 작업 과정상의 주요 차이는 응용 프로그램을 패키지화, 디버깅 및 설치할 때 발생합니다. 예를 들어 AIR for Android는 AIR 패키지 포맷 대신 기본 Android APK 패키지 포맷을 사용합니다. 따라서 표준 Android 설치 및 업데이트 메커니즘도 사용합니다.

AIR for Android

다음은 Android용 AIR 응용 프로그램을 개발할 때 일반적으로 수행하는 단계입니다.

- ActionScript 또는 MXML 코드를 작성합니다.
- AIR 응용 프로그램 설명자 파일을 만듭니다(2.5 이상의 네임스페이스 사용).
- 응용 프로그램을 컴파일합니다.
- 응용 프로그램을 Android 패키지(.apk)로 패키지화합니다.
- 장치 또는 Android 에뮬레이터에 AIR 런타임을 설치합니다(외부 런타임 사용 시. AIR 3.7 이상에서는 전용 런타임이 기본값).
- 장치 또는 Android 에뮬레이터에 응용 프로그램을 설치합니다.
- 장치에서 응용 프로그램을 실행합니다.

Adobe Flash Builder, Adobe Flash Professional CS5 또는 명령줄 도구를 사용하여 이러한 단계를 완료할 수 있습니다.

AIR 응용 프로그램을 완성하고 APK 파일로 패키지화한 후에는 Android Market에 전송하거나 다른 수단을 통해 배포할 수 있습니다.

AIR for iOS

다음은 iOS용 AIR 응용 프로그램을 개발할 때 일반적으로 수행하는 단계입니다.

- iTunes를 설치합니다.
- Apple iOS Provisioning Portal에서 필수 개발자 파일 및 ID를 생성합니다. 그 중 일부는 다음과 같습니다.
  - 개발자 인증서

- 응용 프로그램 ID
- 프로비저닝 프로파일

프로비저닝 프로파일을 만들 때 응용 프로그램을 설치하려는 테스트 장치의 ID를 나열해야 합니다.

- 개발 인증서 및 개인 키를 P12 키 저장소 파일로 변환합니다.
- 응용 프로그램 ActionScript 또는 MXML 코드를 작성합니다.
- ActionScript 또는 MXML 컴파일러를 사용하여 응용 프로그램을 컴파일합니다.
- 응용 프로그램의 아이콘 아트 및 초기 화면 아트를 만듭니다.
- 응용 프로그램 설명자를 만듭니다(2.6 이상의 네임스페이스 사용).
- ADT를 사용하여 IPA 파일을 패키지화합니다.
- iTunes를 사용하여 테스트 장치에 프로비저닝 프로파일을 배치합니다.
- iOS 장치에 응용 프로그램을 설치하고 테스트합니다. iTunes 또는 USB상의 ADT(USB 지원: AIR 3.4 이상)를 사용하여 IPA 파일을 설치할 수 있습니다.

AIR 응용 프로그램이 완성되면 배포 인증서 및 프로비저닝 프로파일을 사용하여 다시 패키지화할 수 있습니다. 그러면 Apple App Store에 전송할 준비가 끝난 것입니다.

모바일 응용 프로그램 속성 설정

다른 AIR 응용 프로그램에서와 마찬가지로 응용 프로그램 설명자 파일에서 기본 응용 프로그램 속성을 설정합니다. 모바일 응용 프로그램에서는 윈도우 크기, 투명도 등 일부 데스크톱 관련 속성이 무시됩니다. 모바일 응용 프로그램은 자체적인 플랫폼 고유의 속성을 사용할 수도 있습니다. 예를 들어 Android 응용 프로그램의 경우 android 요소를, iOS 응용 프로그램의 경우 iPhone 요소를 포함할 수 있습니다.

공통 설정

모든 휴대 장치 응용 프로그램에서 중요한 여러 가지 응용 프로그램 설명자 설정이 있습니다.

필요한 AIR 런타임 버전

응용 프로그램 설명자 파일의 네임스페이스를 사용하여 응용 프로그램에 필요한 AIR 런타임의 버전을 지정합니다.

응용 프로그램에서 사용할 수 있는 기능을 결정하는 데 가장 큰 영향을 미치는 것은 application 요소에 할당된 네임스페이스입니다. 예를 들어 응용 프로그램에서 AIR 2.7 네임스페이스를 사용하는데 사용자에게 그 이후의 버전이 설치되어 있는 경우 응용 프로그램은 AIR 2.7 비헤이비어를 인식하며, 이는 해당 비헤이비어가 이후 버전에서 변경된 경우에도 마찬가지입니다. 네임스페이스를 변경하고 업데이트를 제작할 경우에만 응용 프로그램이 새로운 비헤이비어 및 기능에 액세스하게 됩니다. 보안 픽스는 이 규칙에 대한 중요한 예외입니다.

AIR 3.6 이하에서의 Android와 같이 응용 프로그램과 별도의 런타임을 사용하는 장치에서는 필요한 버전이 없는 경우 AIR를 설치하거나 업그레이드하라는 메시지가 표시됩니다. iPhone과 같이 런타임이 통합된 장치에서는 이러한 상황이 발생하지 않는데, 이는 필요한 버전이 원래부터 응용 프로그램과 함께 패키지화되기 때문입니다.

참고: (AIR 3.7 이상) 기본적으로 ADT는 Android 응용 프로그램과 함께 런타임을 패키지화합니다.

루트 application 요소의 xmlns 특성을 사용하여 네임스페이스를 지정합니다. 다음은 모바일 응용 프로그램에 사용해야 하는 네임스페이스이며, 이는 대상으로 지정한 모바일 플랫폼에 따라 다릅니다.

```
iOS 4+ and iPhone 3Gs+ or Android:
                              <application xmlns="http://ns.adobe.com/air/application/2.7">
                              iOS only:
                              <application xmlns="http://ns.adobe.com/air/application/2.0">
```

참고: iOS 3 장치에 대한 지원은 AIR 2.0 SDK를 기반으로 하는 Packager for iPhone SDK를 통해 제공됩니다. iOS 3용 AIR 응용 프로그램을 만드는 방법에 대한 자세한 내용은 iPhone 응용 프로그램 만들기를 참조하십시오. AIR 2.6 SDK 이상 버전은 iPhone 3Gs, iPhone 4 및 iPad 장치에서 iOS 4 이상을 지원합니다.

기타 도움말 항목

188페이지의 “application”

응용 프로그램 ID

제작하는 각 응용 프로그램에 대해 여러 설정이 고유해야 합니다. 여기에는 ID, 이름 및 파일 이름이 포함됩니다.

Android 응용 프로그램 ID

Android에서 ID는 AIR ID에 접두어 “air”를 붙여서 Android 패키지 이름으로 변환됩니다. 따라서 AIR ID가 **com.example.MyApp**일 경우 Android 패키지 이름은 **air.com.example.MyApp**입니다.

```
<id>com.example.MyApp</id>
                                    <name>My Application</name>
                                    <filename>MyApplication</filename>
```

또한 ID가 Android 운영 체제에서 유효한 패키지 이름이 아닐 경우에는 유효한 이름으로 변환됩니다. 하이픈 문자는 밑줄로 변경되고, ID 구성 요소에 있는 선행 숫자는 앞에 대문자 “A”가 붙습니다. 예를 들어 **3-goats.1-boat**라는 ID는 **air.A3_goats.A1_boat**라는 패키지 이름으로 변환됩니다.

참고: 응용 프로그램 ID에 추가되는 접두어를 사용하여 Android Market에서 AIR 응용 프로그램을 식별할 수 있습니다. 자신의 응용 프로그램이 접두어로 인해 AIR 응용 프로그램으로 식별되는 것을 원치 않는 경우에는 APK 파일의 패키지화를 해제하고, 응용 프로그램 ID를 변경한 다음 Android에 대한 AIR 응용 프로그램 분석 기능 제외에 설명된 대로 다시 패키지화해야 합니다.

iOS 응용 프로그램 ID

Apple iOS Provisioning Portal에서 만든 응용 프로그램 ID와 일치하도록 AIR 응용 프로그램 ID를 설정하십시오.

iOS 응용 프로그램 ID에는 번들 시드 ID가 포함되어 있으며, 번들 시드 ID 뒤에는 번들 식별자가 표시됩니다. 번들 시드 ID는 5RM86Z4DJM과 같이 Apple에서 응용 프로그램 ID에 할당하는 문자열입니다. 번들 식별자에는 자신이 직접 선택하는 역방향 도메인 스타일 이름이 포함됩니다. 번들 식별자는 와일드카드 응용 프로그램 ID를 나타내는 별표(*)로 끝날 수 있습니다. 번들 식별자가 와일드카드 문자로 끝나는 경우에는 해당 와일드카드를 모든 유효한 문자열로 대체할 수 있습니다.

예를 들면 다음과 같습니다.

- Apple 응용 프로그램 ID가 5RM86Z4DJM.com.example.helloWorld인 경우 응용 프로그램 설명자에 com.example.helloWorld를 사용해야 합니다.
- Apple 응용 프로그램 ID가 96LPVWEASL.com.example.*(와일드카드 응용 프로그램 ID)인 경우에는 com.example.helloWorld나 com.example.anotherApp 또는 com.example로 시작되는 다른 ID를 사용할 수 있습니다.
- 마지막으로 Apple 응용 프로그램 ID가 단순히 번들 시드 ID와 와일드카드인 경우(예: 38JE93KJL.*)에는 AIR에서 아무 응용 프로그램 ID나 사용할 수 있습니다.

응용 프로그램 ID를 지정하는 경우 응용 프로그램 ID의 번들 시드 ID 부분은 포함시키지 마십시오.

기타 도움말 항목

203페이지의 “id”

198페이지의 “filename”

211페이지의 “name”

응용 프로그램 버전

AIR 2.5 이상에서는 versionNumber 요소에서 응용 프로그램 버전을 지정합니다. version 요소는 더 이상 사용할 수 없습니다. versionNumber 값을 지정할 때는 최대 세 개의 숫자로 이루어진 시퀀스를 사용해야 하며 이때 각 숫자는 점으로 분리해야 합니다(예: “0.1.2”). 버전 번호의 각 세그먼트는 최대 세 자리까지 가능합니다. 즉, “999.999.999”가 허용되는 가장 큰 버전 번호입니다. 번호에 세 세그먼트를 모두 포함해야 하는 것은 아닙니다. 따라서 “1” 및 “1.0”도 유효한 버전 번호입니다.

versionLabel 요소를 사용하여 버전에 대한 레이블을 지정할 수도 있습니다. 버전 레이블을 추가하면 Android 응용 프로그램 정보 화면 등의 위치에서 버전 번호 대신 표시됩니다. Android Market을 사용하여 배포되는 응용 프로그램에 대해서는 버전 레이블을 지정해야 합니다. AIR 응용 프로그램 설명자에서 versionLabel 값을 지정하지 않으면 versionNumber 값이 [Android 버전 레이블] 필드에 할당됩니다.

```
<!-- AIR 2.5 and later -->
                            <versionNumber>1.23.7<versionNumber>
                            <versionLabel>1.23 Beta 7</versionLabel>
```

Android에서 AIR versionNumber는 Android 정수 versionCode로 변환됩니다. 이때 a*1000000 + b*1000 + c 공식이 사용되는데, 여기서 a, b 및 c는 AIR 버전 번호인 a.b.c의 구성 요소입니다.

기타 도움말 항목

218페이지의 “version”

219페이지의 “versionLabel”

219페이지의 “versionNumber”

기본 응용 프로그램 SWF

initialWindow 요소의 content 자식에서 기본 응용 프로그램 SWF 파일을 지정하십시오. 모바일 프로파일에 있는 장치를 대상으로 삼을 때는 SWF 파일을 사용해야 합니다(HTML 기반 응용 프로그램은 지원되지 않음).

```
<initialWindow>
                            <content>MyApplication.swf</content>
                            </initialWindow>
```

파일을 AIR 패키지에 포함해야 합니다(ADT 또는 IDE 사용). 단순히 응용 프로그램 설명자에 있는 이름을 참조하는 것만으로는 파일이 패키지에 자동으로 포함되지 않습니다.

기본 화면 속성

initialWindow 요소의 여러 자식 요소는 기본 응용 프로그램 화면의 초기 모양 및 비헤이비어를 제어합니다.

- **aspectRatio** - 응용 프로그램이 처음에 세로 형식(높이가 폭보다 큼), 가로 형식(높이가 폭보다 작음) 또는 임의 형식(스테이지에서 모든 방향으로 자동 지정)으로 표시되어야 하는지를 지정합니다.

  ```
  <aspectRatio>landscape</aspectRatio>
  ```

- **autoOrients** - 사용자가 장치를 회전할 때 또는 슬라이딩 키보드를 열거나 닫는 등의 다른 방향 관련 동작을 수행할 때 스테이지의 방향이 자동으로 바뀌는지 여부를 지정합니다. 기본값인 **false**를 사용하면 장치에 맞춰 스테이지의 방향이 변경되지 않습니다.

  ```
  <autoOrients>true</autoOrients>
  ```

- **depthAndStencil** - 심도 또는 스텐실 버퍼를 사용하도록 지정합니다. 일반적으로 이러한 버퍼는 3D 내용으로 작업하는 경우에 사용합니다.

  ```
  <depthAndStencil>true</depthAndStencil>
  ```

- **fullScreen** - 응용 프로그램이 전체 장치 디스플레이를 차지해야 하는지 아니면 시스템 상태 표시줄 등의 일반 운영 체제 크롬과 디스플레이를 공유해야 하는지를 지정합니다.

  ```
  <fullScreen>true</fullScreen>
  ```

- **renderMode** - 런타임이 GPU(Graphics Processing Unit)와 기본 CPU(Central Processing Unit) 중에 어떤 것을 사용하여 응용 프로그램을 렌더링해야 하는지를 지정합니다. 일반적으로 GPU 렌더링을 사용할 때 렌더링 속도가 빠르지만 특정 블렌드 모드와 PixelBender 필터 같은 일부 기능은 GPU 모드에서 사용할 수 없습니다. 또한 각 장치와 각 장치 드라이버는 GPU 기능 및 제한이 서로 다릅니다. GPU 모드를 사용하는 경우 등에는 가능하면 항상 최대한 다양한 장치에서 응용 프로그램을 테스트해야 합니다.

  렌더링 모드는 **gpu**, **cpu**, **direct** 또는 **auto**로 설정할 수 있습니다. 기본값은 현재 CPU 모드로 돌아가는 **auto**입니다.

  참고: 모바일 플랫폼용 AIR와 함께 Flash 내용의 GPU 가속 기능을 활용하려면 renderMode="gpu" 대신 renderMode="direct"(즉, Stage3D)를 사용하는 것이 좋습니다. Adobe에서는 Stage3D 기반 프레임워크인 Starling(2D)과 Away3D(3D)를 공식적으로 지원 및 권장합니다. Stage3D와 Starling/Away3D에 대한 자세한 내용은 http://gaming.adobe.com/getstarted/를 참조하십시오.

  ```
  <renderMode>direct</renderMode>
  ```

  참고: 백그라운드에서 실행되는 응용 프로그램에 대해서는 renderMode="direct"를 사용할 수 없습니다.

  GPU 모드의 제한 사항은 다음과 같습니다.

  - Flex 프레임워크는 GPU 렌더링 모드를 지원하지 않습니다.
  - 필터가 지원되지 않습니다.
  - PixelBender 블렌드 및 채우기가 지원되지 않습니다.
  - 블렌드 모드 중 layer, alpha, erase, overlay, hardlight, lighten 및 darken이 지원되지 않습니다.
  - 비디오를 재생하는 응용 프로그램에서는 GPU 렌더링 모드를 사용하지 않는 것이 좋습니다.
  - GPU 렌더링 모드에서 가상 키보드가 열릴 때 텍스트 필드가 보이는 위치로 제대로 이동하지 않습니다. 사용자가 텍스트를 입력할 때 텍스트 필드가 보이도록 하려면 스테이지의 softKeyboardRect 속성과 소프트 키보드 이벤트를 사용하여 텍스트 필드를 눈에 보이는 영역으로 이동하십시오.
  - GPU를 통해 렌더링할 수 없는 표시 객체는 아예 표시되지 않습니다. 예를 들어 표시 객체에 필터가 적용되어 있으면 해당 객체가 표시되지 않습니다.

  참고: AIR 2.6 이상에서 iOS에 대한 GPU 구현은 이전의 AIR 2.0 버전에서 사용된 구현과 크게 다릅니다. 따라서 서로 다른 최적화 고려 사항이 적용됩니다.

기타 도움말 항목

191페이지의 “aspectRatio”

191페이지의 “autoOrients”

194페이지의 “depthAndStencil”

202페이지의 “fullScreen”

213페이지의 “renderMode”

지원되는 프로파일

supportedProfiles 요소를 추가하여 응용 프로그램에서 지원하는 장치 프로파일을 지정할 수 있습니다. 휴대 장치의 경우 mobileDevice 프로파일을 사용하십시오. ADL(Adobe Debug Launcher)을 사용하여 응용 프로그램을 실행하면 ADL에서 목록에 있는 첫 번째 프로파일을 활성 프로파일로 사용합니다. ADL을 실행할 때 -profile 플래그를 사용하여 지원되는 목록에 있는 특정 프로파일을 선택할 수도 있습니다. 응용 프로그램이 모든 프로파일에서 실행되는 경우에는 supportedProfiles 요소를 전부 제외할 수 있습니다. 이 경우 ADL은 데스크톱 프로파일을 기본 활성 프로파일로 사용합니다.

응용 프로그램이 휴대 장치 및 데스크톱 프로파일을 모두 지원하도록 지정하고 모바일 프로파일에서 응용 프로그램을 테스트하려면 다음 요소를 추가하십시오.

```
<supportedProfiles>mobileDevice desktop</supportedProfiles>
```

기타 도움말 항목

216페이지의 “supportedProfiles”

222페이지의 “장치 프로파일”

144페이지의 “ADL(AIR Debug Launcher)”

필수 기본 확장

mobileDevice 프로파일을 지원하는 응용 프로그램에서는 기본 확장을 사용할 수 있습니다.

응용 프로그램 설명자에서 AIR 응용 프로그램이 사용하는 모든 기본 확장을 선언합니다. 다음 예제에서는 두 개의 필수 기본 확장을 지정하기 위한 구문을 보여 줍니다.

```
<extensions>
                                <extensionID>com.example.extendedFeature</extensionID>
                                <extensionID>com.example.anotherFeature</extensionID>
                                </extensions>
```

extensionID 요소의 값은 확장 설명자 파일에서 id 요소의 값과 동일합니다. 확장 설명자 파일은 extension.xml이라는 XML 파일로, 이 파일은 기본 확장 개발자로부터 받는 ANE 파일에 패키지화되어 있습니다.

가상 키보드 비헤이비어

가상 키보드가 나타난 후에 포커스가 있는 텍스트 입력 필드가 표시되도록 하기 위해 런타임이 사용하는 자동 패닝 및 크기 조절 비헤이비어를 사용하지 않으려면 softKeyboardBehavior 요소를 none으로 설정하십시오. 자동 비헤이비어를 사용하지 않는 경우 키보드가 나타난 후에 텍스트 입력 영역 또는 기타 관련 내용이 표시되도록 하는 동작은 전적으로 응용 프로그램에서 수행합니다. 스테이지의 softKeyboardRect 속성을 SoftKeyboardEvent와 함께 사용하여 키보드가 열릴 때를 감지하고 이로 인해 가려지는 영역을 확인할 수 있습니다.

자동 비헤이비어를 사용하려면 이 요소 값을 pan으로 설정하십시오.

```
<softKeyboardBehavior>pan</softKeyboardBehavior>
```

pan이 기본값이기 때문에 softKeyboardBehavior 요소를 생략해도 자동 키보드 비헤이비어가 사용됩니다.

참고: GPU 렌더링도 사용하는 경우에는 팬 비헤이비어가 지원되지 않습니다.

기타 도움말 항목

215페이지의 “softKeyboardBehavior”

Stage.softKeyboardRect

SoftKeyboardEvent

Android 설정

Android 플랫폼에서 응용 프로그램 설명자의 android 요소를 사용하여 Android 응용 프로그램 매니페스트에 정보를 추가할 수 있습니다. Android 응용 프로그램 매니페스트는 Android 운영 체제에서 사용하는 응용 프로그램 속성 파일입니다. ADT에서는 APK 패키지를 만들 때 Android Manifest.xml 파일을 자동으로 생성합니다. AIR는 특정 기능이 작동하는 데 필요한 값에 몇 가지 속성을 설정합니다. AIR 응용 프로그램 설명자 파일의 android 섹션에 설정된 다른 모든 속성은 Manifest.xml 파일의 해당 섹션에 추가됩니다.

참고: 대부분의 AIR 응용 프로그램에서는 응용 프로그램에 필요한 Android 권한을 android 요소 내에서 설정해야 하지만, 일반적으로 다른 속성은 설정할 필요가 없습니다.

문자열, 정수 또는 부울 값을 가지는 특성만 설정할 수 있습니다. 응용 프로그램 패키지에 있는 리소스에 대한 참조를 설정하는 것은 지원되지 않습니다.

참고: 런타임에는 14 이상의 최소 SDK 버전이 필요합니다. 따라서 더 높은 버전용으로만 응용 프로그램을 만들려는 경우에는 Manifest에 <uses-sdk android:minSdkVersion=""></uses-sdk>가 올바른 버전으로 포함되는지 확인해야 합니다.

예약된 Android 매니페스트 설정

AIR는 응용 프로그램 및 런타임 기능이 올바르게 작동하도록 하기 위해 생성된 Android 매니페스트 문서에서 여러 매니페스트 항목을 설정합니다. 다음 설정은 정의할 수 없습니다.

manifest 요소

manifest 요소의 다음 특성은 설정할 수 없습니다.

- package
- android:versionCode
- android:versionName
- xmlns:android

activity 요소

기본 activity 요소의 다음 특성은 설정할 수 없습니다.

- android:label
- android:icon

application 요소

application 요소의 다음 특성은 설정할 수 없습니다.

- android:theme
- android:name
- android:label
- android:windowSoftInputMode
- android:configChanges
- android:screenOrientation
- android:launchMode

Android 권한

Android 보안 모델에서는 보안 또는 개인 정보 보호와 관련된 기능을 사용할 경우 각 응용 프로그램이 권한을 요청하도록 요구하고 있습니다. 이러한 권한은 응용 프로그램이 패키지화될 때 지정되어야 하며, 런타임에 변경될 수 없습니다. Android 운영 체제에서는 사용자가 응용 프로그램을 설치할 때 해당 응용 프로그램이 어떤 권한을 요청하는지를 알려 줍니다. 기능에 필요한 권한이 요청되지 않는 경우 Android 운영 체제에서는 응용 프로그램이 해당 기능에 액세스할 때 예외가 발생할 수도 있지만, 항상 그런 것은 아닙니다. 예외는 런타임을 통해 응용 프로그램으로 전송됩니다. 감지되지 않는 오류의 경우에는 Android 시스템 로그에 권한 오류 메시지가 추가됩니다.

AIR에서는 응용 프로그램 설명자의 android 요소 내에 Android 권한을 지정합니다. 권한을 추가할 때는 다음 포맷이 사용됩니다(여기서 PERMISSION_NAME은 Android 권한의 이름).

```
<android>
                                <manifestAdditions>
                                <![CDATA[
                                <manifest>
                                <uses-permission android:name="android.permission.PERMISSION_NAME" />
                                </manifest>
                                ]]>
                                </manifestAdditions>
                                </android>
```

manifest 요소 내의 uses-permissions 문은 Android 매니페스트 문서에 바로 추가됩니다.

다양한 AIR 기능을 사용하려면 다음 권한이 필요합니다.

ACCESS_COARSE_LOCATION 응용 프로그램에서 Geolocation 클래스를 통해 WIFI 및 셀룰러 네트워크 위치 데이터에 액세스할 수 있도록 허용합니다.

ACCESS_FINE_LOCATION 응용 프로그램에서 Geolocation 클래스를 통해 GPS 데이터에 액세스할 수 있도록 허용합니다.

ACCESS_NETWORK_STATE 및 ACCESS_WIFI_STATE 응용 프로그램에서 NetworkInfo 클래스를 통해 네트워크 정보에 액세스할 수 있도록 허용합니다.

CAMERA 응용 프로그램에서 카메라에 액세스할 수 있도록 허용합니다.

참고: 카메라 기능을 사용하기 위한 권한을 요청하면 Android에서는 응용 프로그램도 카메라를 필요로 하는 것으로 가정합니다. 카메라가 응용 프로그램의 선택적 기능인 경우에는 uses-feature 요소를 카메라의 매니페스트에 추가하여 필수 특성을 false로 설정해야 합니다. 72페이지의 “Android 호환성 필터링”을 참조하십시오.

INTERNET 응용 프로그램에서 네트워크 요청을 수행할 수 있도록 허용하고 원격 디버깅도 허용합니다.

READ_PHONE_STATE AIR 런타임에서 전화 통화 중에 오디오를 음소거할 수 있도록 허용합니다. 응용 프로그램이 백그라운드에서 오디오를 재생하는 경우 이 권한을 설정해야 합니다.

RECORD_AUDIO 응용 프로그램에서 마이크에 액세스할 수 있도록 허용합니다.

WAKE_LOCK 및 DISABLE_KEYGUARD 응용 프로그램에서 장치가 SystemIdleMode 클래스 설정을 사용하여 대기 모드로 들어가지 못하게 할 수 있도록 허용합니다.

WRITE_EXTERNAL_STORAGE 응용 프로그램에서 장치의 외부 메모리 카드에 쓸 수 있도록 허용합니다.

예를 들어 모든 권한을 요구하는 응용 프로그램에 대한 권한을 설정하려면 응용 프로그램 설명자에 다음을 추가하면 됩니다.

```
<android>
                        <manifestAdditions>
                        <![CDATA[
                        <manifest>
                        <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
                        <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
                        <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
                        <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
                        <uses-permission android:name="android.permission.CAMERA" />
                        <uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
                        <uses-permission android:name="android.permission.INTERNET" />
                        <uses-permission android:name="android.permission.READ_PHONE_STATE" />
                        <uses-permission android:name="android.permission.RECORD_AUDIO" />
                        <uses-permission android:name="android.permission.WAKE_LOCK" />
                        <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
                        </manifest>
                        ]]>
                        </manifestAdditions>
                        </android>
```

기타 도움말 항목

Android 보안 및 권한

Android Manifest.permission 클래스

Android 사용자 정의 URI 스킴

사용자 정의 URI 스킴을 사용하여 웹 페이지 또는 기본 Android 응용 프로그램에서 AIR 응용 프로그램을 실행할 수 있습니다. 사용자 정의 URI 지원은 Android 매니페스트에 지정된 intent filter를 사용하므로 이 기술을 다른 플랫폼에서 사용할 수는 없습니다.

사용자 정의 URI를 사용하려면 <android> 블록 내에서 응용 프로그램 설명자에 intent-filter를 추가하십시오. 다음 예제의 두 intent-filter 요소를 모두 지정해야 합니다. 사용자 정의 스킴의 URI 문자열이 반영되도록 <data android:scheme="***my-customuri***"/> 문을 편집합니다.

```
<android>
                        <manifestAdditions>
                        <![CDATA[
                        <manifest>
                        <application>
                        <activity>
                        <intent-filter>
                        <action android:name="android.intent.action.MAIN"/>
                        <category android:name="android.intent.category.LAUNCHER"/>
                        </intent-filter>
                        <intent-filter>
                        <action android:name="android.intent.action.VIEW"/>
                        <category android:name="android.intent.category.BROWSABLE"/>
                        <category android:name="android.intent.category.DEFAULT"/>
                        <data android:scheme="my-customuri"/>
                        </intent-filter>
                        </activity>
                        </application>
                        </manifest>
                        ]]>
                        </manifestAdditions>
                        </android>
```

intent filter는 해당 응용 프로그램을 사용하여 특정 작업을 수행할 수 있음을 Android 운영 체제에 알려 줍니다. 사용자 정의 URI의 경우 이는 사용자가 해당 URI 스킴을 사용하는 링크를 클릭했다는 의미입니다(그리고 브라우저에서는 이를 처리하는 방법을 인식하지 못함).

사용자 정의 URI를 통해 응용 프로그램을 호출할 때는 NativeApplication 객체가 invoke 이벤트를 전달합니다. 쿼리 매개 변수를 비롯한 링크의 URL은 InvokeEvent 객체의 arguments 배열에 배치됩니다. intent-filter는 수에 관계없이 사용할 수 있습니다.

참고: StageWebView 인스턴스의 링크는 사용자 정의 URI 스킴을 사용하는 URL을 열 수 없습니다.

기타 도움말 항목

Android intent filter

Android 작업 및 범주

Android 호환성 필터링

Android 운영 체제는 응용 프로그램 매니페스트 파일에서 다수의 요소를 사용하여 응용 프로그램이 특정 장치와 호환되는지 여부를 확인합니다. 이 정보를 매니페스트에 추가하는 것은 선택 사항입니다. 이러한 요소를 포함하지 않으면 응용 프로그램이 모든 Android 장치에 설치될 수 있습니다. 하지만 어떠한 Android 장치에서도 제대로 작동하지 않을 수 있습니다. 예를 들어 카메라가 없는 휴대폰에서는 카메라 응용 프로그램이 크게 유용하지 않을 것입니다.

필터링에 사용할 수 있는 Android 매니페스트 태그는 다음과 같습니다.

- supports-screens
- uses-configuration
- uses-feature
- uses-sdk(AIR 3 이상)

카메라 응용 프로그램

응용 프로그램에 대해 카메라 권한을 요청하는 경우 Android는 자동 포커스, 플래시 등 사용 가능한 모든 카메라 기능이 응용 프로그램에 필요하다고 가정합니다. 응용 프로그램에서 모든 카메라 기능이 필요하지 않거나 카메라가 선택적인 기능일 경우에는 카메라의 다양한 uses-feature 요소를 설정하여 선택 사항임을 나타내야 합니다. 이렇게 하지 않으면 특정 기능이 없거나 카메라가 전혀 없는 장치를 보유한 사용자들이 Android Market에서 귀하의 응용 프로그램을 찾을 수 없습니다.

다음 예제에서는 카메라에 대한 권한을 요청하고 모든 카메라 기능을 선택 사항으로 만드는 방법을 보여 줍니다.

```
<android>
                                <manifestAdditions>
                                <![CDATA[
                                <manifest>
                                <uses-permission android:name="android.permission.CAMERA" />
                                <uses-feature android:name="android.hardware.camera"
android:required="false"/>
                                <uses-feature android:name="android.hardware.camera.autofocus"
android:required="false"/>
                                <uses-feature android:name="android.hardware.camera.flash"
android:required="false"/>
                                </manifest>
                                ]]>
                                </manifestAdditions>
                                </android>
```

오디오 녹음 응용 프로그램

오디오를 녹음하기 위한 권한을 요청할 경우 Android에서는 응용 프로그램에 마이크도 필요한 것으로 가정합니다. 오디오 녹음이 응용 프로그램의 선택적 기능일 경우 uses-feature 태그를 사용하여 마이크가 필수가 아님을 지정할 수 있습니다. 이렇게 하지 않으면 마이크가 없는 장치의 사용자들이 Android Market에서 귀하의 응용 프로그램을 찾을 수 없습니다.

다음 예제에서는 마이크 하드웨어를 선택 사항으로 지정한 상태에서 마이크 사용 권한을 요청하는 방법을 보여 줍니다.

```
<android>
                                <manifestAdditions>
                                <![CDATA[
                                <manifest>
                                <uses-permission android:name="android.permission.RECORD_AUDIO" />
                                <uses-feature android:name="android.hardware.microphone"
android:required="false"/>
                                </manifest>
                                ]]>
                                </manifestAdditions>
                                </android>
```

기타 도움말 항목

Android 개발자: Android 호환성

Android 개발자: Android 기능 이름 상수

설치 위치

Android manifest 요소의 installLocation 특성을 auto 또는 preferExternal로 설정함으로써 응용 프로그램을 외부 메모리 카드에 설치하거나 외부 메모리 카드로 이동하는 것을 허용할 수 있습니다.

```
<android>
                            <manifestAdditions>
                            <![CDATA[
                            <manifest android:installLocation="preferExternal"/>
                            ]]>
                            </manifestAdditions>
                            </android>
```

Android 운영 체제에서는 응용 프로그램이 외부 메모리에 확실히 설치된다고 보장하지 않습니다. 사용자는 시스템 설정 응용 프로그램을 사용하여 내부 및 외부 메모리 간에 응용 프로그램을 이동할 수도 있습니다.

외부 메모리에 설치된 경우에도 app-storage 디렉토리의 내용, 공유 객체, 임시 파일 등의 응용 프로그램 캐시 및 사용자 데이터는 여전히 내부 메모리에 저장됩니다. 내부 메모리를 너무 많이 사용하지 않도록 하려면 응용 프로그램 저장소 디렉토리에 저장할 데이터를 신중하게 선택하십시오. File.userDirectory 또는 File.documentsDirectory 위치(둘 다 Android에서 SD 카드의 루트에 매핑됨)를 사용하여 많은 양의 데이터를 SDCard에 저장해야 합니다.

Flash Player 및 StageWebView 객체의 다른 플러그인 활성화

Android 3.0 이상의 경우 StageWebView 객체의 플러그인 내용을 표시하려면 응용 프로그램이 Android 응용 프로그램 요소에서 하드웨어 가속화를 활성화해야 합니다. 플러그인 렌더링을 사용하려면 application 요소의 android:hardwareAccelerated 특성을 true로 설정합니다.

```
<android>
                            <manifestAdditions>
                            <![CDATA[
                            <manifest>
                            <application android:hardwareAccelerated="true"/>
                            </manifest>
                            ]]>
                            </manifestAdditions>
                            </android>
```

색상 심도
AIR 3 이상

AIR 3 이상에서는 런타임이 디스플레이에서 32비트 색상을 렌더링하도록 설정합니다. 이전 버전의 AIR에서는 런타임이 16비트 색상을 사용합니다. 응용 프로그램 설명자의 <colorDepth> 요소를 사용하여 런타임이 16비트 색상을 사용하도록 할 수도 있습니다.

```
<android>
                            <colorDepth>16bit</colorDepth>
                            <manifestAdditions>...</manifestAdditions>
                            </android>
```

16비트 색상 심도를 사용하면 렌더링 성능이 증가하지만 색상 충실도가 떨어질 수 있습니다.

iOS 설정

iOS 장치에만 적용되는 설정은 응용 프로그램 설명자에서 <iPhone> 요소 안에 있습니다. iPhone 요소는 자식 요소로 InfoAdditions 요소, requestedDisplayResolution 요소, Entitlements 요소, externalSwfs 요소 및 forceCPURenderModeForDevices 요소를 보유할 수 있습니다.

InfoAdditions 요소를 사용하여 응용 프로그램의 Info.plist 설정 파일에 추가되는 키-값 쌍을 지정할 수 있습니다. 예를 들어 다음 값은 응용 프로그램의 상태 표시줄 스타일을 설정하며, 응용 프로그램에 지속적인 Wi-Fi 액세스가 필요하지 않음을 나타냅니다.

```
<InfoAdditions>
                            <![CDATA[
                            <key>UIStatusBarStyle</key>
                            <string>UIStatusBarStyleBlackOpaque</string>
                            <key>UIRequiresPersistentWiFi</key>
                            <string>NO</string>
                            ]]>
                            </InfoAdditions>
```

InfoAdditions 설정은 CDATA 태그 안에 있습니다.

Entitlements 요소를 사용하면 응용 프로그램에 대한 Entitlements.plist 설정 파일에 추가되는 키-값 쌍을 지정할 수 있습니다. Entitlements.plist 설정에서는 응용 프로그램이 푸시 알림과 같은 특정 iOS 기능에 액세스할 수 있도록 지원합니다.

Info.plist 및 Entitlements.plist 설정에 대한 자세한 내용은 Apple 개발자 설명서를 참조하십시오.

iOS에서 백그라운드 작업 지원
AIR 3.3

Adobe AIR 3.3 이상은 특정 백그라운드 비헤이비어를 사용하여 iOS에서 멀티태스킹을 지원합니다.

- 오디오
- 위치 업데이트
- 네트워킹
- 백그라운드 응용 프로그램 실행 옵트아웃

참고: swf 버전 21 이하에서는 renderMode가 direct로 설정된 경우 AIR가 iOS 및 Android에서 백그라운드 실행을 지원하지 않습니다. 이러한 제한으로 인해 Stage3D 기반 응용 프로그램이 오디오 재생, 위치 업데이트, 네트워크 업로드/다운로드 등의 백그라운드 작업을 실행할 수 없습니다. iOS는 백그라운드에서 OpenGLES 또는 호출의 렌더링을 허용하지 않습니다. 백그라운드에서 OpenGL 호출을 수행하려는 응용 프로그램은 iOS에 의해 종료됩니다. Android는 응용 프로그램이 백그라운드에서 OpenGLES 호출을 수행하거나 오디오 재생과 같은 다른 백그라운드 작업을 수행하는 것을 제한하지 않습니다. swf 버전 22 이

상에서는 renderMode가 direct로 설정된 경우 AIR 모바일 응용 프로그램이 백그라운드에서 실행될 수 있습니다. 백그라운드에서 OpenGLES 호출을 수행하는 경우 AIR iOS 런타임에서 ActionScript 오류(3768 - 백그라운드 실행 중 Stage3D API를 사용할 수 없음)가 발생합니다. 그러나 Android에서는 기본 응용 프로그램이 백그라운드에서 OpenGLES 호출을 수행할 수 있으므로 오류가 발생하지 않습니다. 모바일 리소스의 최적 활용을 위해 백그라운드에서 응용 프로그램이 실행되는 동안에는 렌더링 호출을 수행하지 마십시오.

백그라운드 오디오

백그라운드 오디오 재생 및 녹음을 사용하려면 InfoAdditions 요소에 다음 키-값 쌍을 포함합니다.

```
<InfoAdditions>
                                <![CDATA[
                                <key>UIBackgroundModes</key>
                                <array>
                                <string>audio</string>
                                </array>
                                ]]>
                                </InfoAdditions>
```

백그라운드 위치 업데이트

백그라운드 위치 업데이트를 사용하려면 InfoAdditions 요소에 다음 키-값 쌍을 포함합니다.

```
<InfoAdditions>
                                <![CDATA[
                                <key>UIBackgroundModes</key>
                                <array>
                                <string>location</string>
                                </array>
                                ]]>
                                </InfoAdditions>
```

참고: 위치 API는 배터리 전원을 많이 소모하므로 이 기능은 필요한 경우에만 사용하십시오.

백그라운드 네트워킹

백그라운드에서 짧은 작업을 실행하기 위해 응용 프로그램에서는 NativeApplication.nativeApplication.executeInBackground 속성을 true로 설정합니다.

예를 들어 응용 프로그램에서 파일 업로드 작업을 시작한 후 사용자가 다른 응용 프로그램을 전면으로 이동할 수 있습니다. 응용 프로그램이 업로드 완료 이벤트를 수신하면 NativeApplication.nativeApplication.executeInBackground를 false로 설정할 수 있습니다.

iOS는 백그라운드 작업에 시간 제한을 적용하므로 NativeApplication.nativeApplication.executeInBackground 속성을 true로 설정해도 응용 프로그램이 무기한으로 실행되지는 않습니다. iOS가 백그라운드 처리를 중지하면 AIR에서 NativeApplication.suspend 이벤트를 전달합니다.

백그라운드 실행 옵트아웃

응용 프로그램에서 InfoAdditions 요소에 다음 키-값 쌍을 포함하여 백그라운드 실행에서 명시적으로 옵트아웃할 수 있습니다.

```
<InfoAdditions>
                                <![CDATA[
                                <key>UIApplicationExitsOnSuspend</key>
                                <true/>
                                ]]>
                                </InfoAdditions>
```

예약된 iOS InfoAdditions 설정

AIR는 응용 프로그램 및 런타임 기능이 올바르게 작동하도록 하기 위해 생성된 Info.plist 파일에서 여러 항목을 설정합니다. 다음 설정은 정의할 수 없습니다.

CFBundleDisplayName	CTInitialWindowTitle
CFBundleExecutable	CTInitialWindowVisible
CFBundleIconFiles	CTIosSdkVersion
CFBundleIdentifier	CTMaxSWFMajorVersion
CFBundleInfoDictionaryVersion	DTPlatformName
CFBundlePackageType	DTSDKName
CFBundleResourceSpecification	MinimumOSVersion (3.2까지 예약됨)
CFBundleShortVersionString	NSMainNibFile
CFBundleSupportedPlatforms	UIInterfaceOrientation
CFBundleVersion	UIStatusBarHidden
CTAutoOrients	UISupportedInterfaceOrientations

참고: MinimumOSVersion을 정의할 수 있습니다. MinimumOSVersion 정의는 Air 3.3 이상에서 인정됩니다.

다른 iOS 장치 모델 지원

iPad 지원의 경우 UIDeviceFamily에 대한 올바른 키-값 설정을 InfoAdditions 요소 안에 포함하십시오. UIDeviceFamily 설정은 문자열 배열입니다. 각 문자열은 지원되는 장치를 정의합니다. <string>1</string> 설정은 iPhone 및 iPod Touch에 대한 지원을 정의합니다. <string>2</string> 설정은 iPad에 대한 지원을 정의합니다. <string>3</string> 설정은 tvOS에 대한 지원을 정의합니다. 이러한 문자열 중 하나만 지정하는 경우 해당 장치 제품군만 지원됩니다. 예를 들어 다음 설정은 iPad로 지원을 제한합니다.

```
<key>UIDeviceFamily</key>
                              <array>
                              <string>2</string>
                              </array>>
```

다음 설정은 두 가지 장치 제품군(iPhone/iPod Touch 및 iPad)을 모두 지원합니다.

```
<key>UIDeviceFamily</key>
                              <array>
                              <string>1</string>
                              <string>2</string>
                              </array>
```

또한 AIR 3.7 이상에서 forceCPURenderModeForDevices 태그를 사용하여, 지정한 장치 집합에 대해 CPU 렌더링 모드를 강제 적용하고 나머지 iOS 장치에 대해 GPU 렌더링 모드를 사용하도록 설정할 수 있습니다.

이 태그는 iPhone 태그의 자식으로 추가하고, 장치 모델 이름의 목록은 공백으로 구분하여 지정합니다. 유효한 장치 모델 이름 목록은 201페이지의 “forceCPURenderModeForDevices”를 참조하십시오.

예를 들어 기존 iPod, iPhone 및 iPad에 대해 CPU 모드를 사용하고 다른 모든 장치에 대해 GPU 모드를 사용하려면 응용 프로그램 설명자에서 다음을 지정하면 됩니다.

```
...
                              <renderMode>GPU</renderMode>
                              ...
                              <iPhone>
                              ...
                                 <forceCPURenderModeForDevices>iPad1,1 iPhone1,1 iPhone1,2 iPod1,1
                                 </forceCPURenderModeForDevices>
                              </iPhone>
```

고해상도 디스플레이

requestedDisplayResolution 요소는 응용 프로그램이 고해상도 스크린이 있는 iOS 장치에서 **standard** 모드를 사용해야 하는지 아니면 **high** 해상도 모드를 사용해야 하는지를 지정합니다.

```
<requestedDisplayResolution>high</requestedDisplayResolution>
```

고해상도 모드에서는 고해상도 디스플레이의 각 픽셀을 개별적으로 처리할 수 있습니다. 표준 모드에서는 응용 프로그램의 장치 스크린이 표준 해상도 스크린으로 나타납니다. 이 모드에서 한 픽셀을 그리면 고해상도 스크린에 있는 4개 픽셀의 색상이 설정됩니다.

기본 설정은 standard입니다. iOS 장치를 대상으로 할 때는 requestedDisplayResolution 요소를 iPhone 요소의 자식으로 사용합니다(InfoAdditions 요소 또는 initialWindow 요소가 아님).

장치마다 다른 설정을 사용하려면 기본값을 requestedDisplayResolution 요소의 값으로 지정합니다. 반대되는 값을 사용해야 하는 장치를 지정할 때는 excludeDevices 특성을 사용합니다. 예를 들어 다음 코드에서 표준 모드를 사용하는 3세대 iPad를 제외하고는 고해상도 모드를 지원하는 모든 장치에서 고해상도 모드가 사용됩니다.

```
<requestedDisplayResolution excludeDevices="iPad3">high</requestedDisplayResolution>
```

excludeDevices 특성은 AIR 3.6 이상에서 사용할 수 있습니다.

기타 도움말 항목

214페이지의 “requestedDisplayResolution”

Renaun Erickson: AIR 2.6을 사용하여 Retina 및 Retina가 아닌 iOS 화면 모두에 적합하도록 개발

iOS 사용자 정의 URI 스킴

웹 페이지의 링크에 의해 또는 장치에 있는 다른 기본 응용 프로그램에 의해 응용 프로그램이 호출될 수 있도록 사용자 정의 URI 스킴을 등록할 수 있습니다. URI 스킴을 등록하려면 CFBundleURLTypes 키를 InfoAdditions 요소에 추가하십시오. 다음 예제는 **com.example.app**이라는 URI 스킴을 등록하여 응용 프로그램이 **example://foo** 형태의 URL을 통해 호출될 수 있도록 합니다.

```
<key>CFBundleURLTypes</key>
                              <array>
                              <dict>
                              <key>CFBundleURLSchemes</key>
                              <array>
                              <string>example</string>
                              </array>
                              <key>CFBundleURLName</key>
                              <string>com.example.app</string>
                              </dict>
                              </array>
```

사용자 정의 URI를 통해 응용 프로그램을 호출할 때는 NativeApplication 객체가 invoke 이벤트를 전달합니다. 쿼리 매개 변수를 비롯한 링크의 URL은 InvokeEvent 객체의 arguments 배열에 배치됩니다. 사용자 정의 URI 스킴은 개수에 제한 없이 사용할 수 있습니다.

참고: StageWebView 인스턴스의 링크는 사용자 정의 URI 스킴을 사용하는 URL을 열 수 없습니다.

참고: 다른 응용 프로그램이 스킴을 이미 등록한 경우에는 자신의 응용 프로그램이 해당 URI 스킴에 대해 등록된 응용 프로그램으로 대체될 수 없습니다.

iOS 호환성 필터링

특정 하드웨어 또는 소프트웨어 기능이 있는 장치에서만 응용 프로그램을 사용해야 하는 경우 InfoAdditions 요소 내에서 UIRequiredDeviceCapabilities 배열에 항목을 추가하십시오. 예를 들어 다음 항목은 응용 프로그램에 스틸 카메라 및 마이크가 필요하다는 것을 나타냅니다.

```
<key>UIRequiredDeviceCapabilities</key>
                                <array>
                                <string>microphone</string>
                                <string>still-camera</string>
                                </array>
```

해당 기능이 장치에 없는 경우 응용 프로그램을 설치할 수 없습니다. AIR 응용 프로그램과 관련된 기능 설정은 다음과 같습니다.

telephony	camera-flash
wifi	video-camera
sms	accelerometer
still-camera	location-services
auto-focus-camera	gps
front-facing-camera	microphone

AIR 2.6 이상은 **armv7** 및 **opengles-2**를 필수 기능 목록에 자동으로 추가합니다.

참고: 응용 프로그램 설명자에 이들 기능을 포함하지 않아도 응용 프로그램에서는 이러한 기능을 사용할 수 있습니다. 응용 프로그램이 제대로 작동하지 않는 장치에 사용자가 응용 프로그램을 설치하는 것을 방지하기 위한 목적으로만 UIRequiredDeviceCapabilities 설정을 사용하십시오.

일시 중지 대신 종료

사용자가 AIR 응용 프로그램에서 전환하여 밖으로 나오면 AIR 응용 프로그램이 백그라운드로 들어가고 일시 중지됩니다. 응용 프로그램이 일시 중지되는 대신 완전히 종료되도록 하려면 UIApplicationExitsOnSuspend 속성을 YES로 설정하십시오.

```
<key>UIApplicationExitsOnSuspend</key>
                                <true/>
```

외부 에셋 전용 SWF를 로드하여 다운로드 크기 최소화

AIR 3.7

응용 프로그램에 사용되는 하위 집합의 SWF를 패키지화하고 Loader.load() 메서드를 사용하여 나머지 외부(에셋 전용) SWF를 런타임에 로드하여 초기 응용 프로그램 다운로드 크기를 최소화할 수 있습니다. 이 기능을 사용하려면 ADT가 외부 로드된 SWF 파일에서 기본 응용 프로그램 SWF로 모든 ActionScript ByteCode(ABC)를 이동할 수 있도록 응용 프로그램을 패키지화해야 합니다. 그래야 SWF에는 에셋만 포함됩니다. 이는 응용 프로그램을 설치한 후 어떠한 코드 다운로드도 금지하는 Apple Store의 규칙을 준수합니다.

ADT는 외부에서 로드된 SWF(일명, 누락된 SWF)를 지원하기 위해 다음을 수행합니다.

- <iPhone> 요소의 <externalSwfs> 하위 요소에 지정된 텍스트 파일을 읽고 행으로 구분된 SWF 목록에 액세스하여 실행 시 로드합니다.

  ```
  <iPhone>
                                                  . . .
      <externalSwfs>FilewithPathsOfSWFsThatAreToNotToBePackaged.txt</externalSwfs>
                                              </iPhone>
  ```

- 외부에서 로드된 SWF에서 기본 실행 파일로 ABC 코드를 전송합니다.
- 외부에서 로드된 SWF 파일을 .ipa 파일에서 누락시킵니다.
- 누락된 SWF를 .remoteStrippedSWFs 디렉토리에 복사합니다. 이러한 SWF는 웹 서버에 호스팅하고 필요한 경우 응용 프로그램이 런타임에 해당 SWF를 로드합니다.

다음 예에 나타난 것처럼 런타임에 로드할 SWF 파일은 텍스트 파일에서 한 줄에 하나씩 이름을 지정하여 나타내야 합니다.

```
assets/Level1/Level1.swf
                            assets/Level2/Level2.swf
                            assets/Level3/Level3.swf
                            assets/Level4/Level4.swf
```

지정한 파일 경로는 응용 프로그램 설명자 파일에 대한 상대 경로입니다. 또한 adt 명령에서 이러한 SWF를 에셋으로 지정해야 합니다.

참고: 이러한 기능은 표준 패키징에만 적용됩니다. 인터프리터, 시뮬레이터, 디버그 등을 사용하는 빠른 패키징의 경우 ADT는 누락된 SWF를 만들지 않습니다.

샘플 코드를 포함하여 이 기능에 대한 자세한 내용은 Adobe 엔지니어인 Abhinav Dhandh가 작성한 블로그 게시물 iOS에서 AIR 응용 프로그램을 위해 보조 SWF 외부 호스팅을 참조하십시오.

Geolocation 지원

Geolocation 지원을 위해서는 다음 키-값 쌍 중 하나를 InfoAdditions 요소에 추가하십시오.

```
<InfoAdditions>
                            <![CDATA[
                            <key>NSLocationAlwaysUsageDescription</key>
                            <string>Sample description to allow geolocation always</string>
                            <key>NSLocationWhenInUseUsageDescription</key>
                            <string>Sample description to allow geolocation when application is in
foreground</string>
                            ]]>
                            </InfoAdditions>
```

응용 프로그램 아이콘

다음 표에는 각 모바일 플랫폼에 사용되는 아이콘 크기가 나와 있습니다.

아이콘 크기	플랫폼
29x29	iOS
36x36	Android
40x40	iOS
48x48	Android, iOS
50x50	iOS
57x57	iOS
58x58	iOS
60x60	iOS
72x72	Android, iOS
75x75	iOS
76x76	iOS
80x80	iOS
87x87	iOS
96x96	Android
100x100	iOS

아이콘 크기	플랫폼
114x114	iOS
120x120	iOS
144x144	Android, iOS
152x152	iOS
167x167	iOS
180x180	iOS
192x192	Android
512x512	Android, iOS
1024x1024	iOS

응용 프로그램 설명자 파일의 아이콘 요소에서 아이콘 파일의 경로를 지정합니다.

```
<icon>
                        <image36x36>assets/icon36.png</image36x36>
                        <image48x48>assets/icon48.png</image48x48>
                        <image72x72>assets/icon72.png</image72x72>
                        </icon>
```

지정된 크기의 아이콘을 제공하지 않으면 그 다음으로 가장 큰 아이콘이 지정된 크기에 맞게 조정되어 사용됩니다.

Android의 아이콘

Android에서 응용 프로그램 설명자로 지정된 아이콘은 응용 프로그램 실행 아이콘으로 사용됩니다. 응용 프로그램 실행 아이콘은 36x36, 48x48, 72x72, 96x96, 144x144 및 192x192 픽셀 PNG 이미지 집합으로 제공되어야 합니다. 이러한 아이콘 크기는 각각 낮은 밀도, 보통 밀도 및 높은 밀도의 화면에 사용됩니다.

개발자는 Google Play Store에 앱을 제출할 때 512x512 픽셀 아이콘을 제출해야 합니다.

iOS의 아이콘

응용 프로그램 설명자에 정의된 아이콘은 iOS 응용 프로그램의 다음 위치에서 사용됩니다.

- 29x29 픽셀 아이콘 - 저해상도 iPhone/iPod의 내부 검색 아이콘 및 저해상도 iPad의 설정 아이콘입니다.
- 40x40 픽셀 아이콘 - 저해상도 iPad의 내부 검색 아이콘입니다.
- 48x48 픽셀 아이콘 - AIR에서 이 이미지에 테두리를 추가하여 저해상도 iPad의 내부 검색에 대한 50x50 아이콘으로 사용합니다.
- 50x50 픽셀 아이콘 - 저해상도 iPad의 내부 검색 아이콘입니다.
- 57x57 픽셀 아이콘 - 저해상도 iPhone/iPod의 응용 프로그램 아이콘입니다.
- 58x58 픽셀 아이콘 - Retina 디스플레이 iPhone/iPod의 내부 검색 아이콘 및 Retina 디스플레이 iPad의 설정 아이콘입니다.
- 60x60 픽셀 아이콘 - 저해상도 iPhone/iPod의 응용 프로그램 아이콘입니다.
- 72x72 픽셀 아이콘(선택 사항) - 저해상도 iPad의 응용 프로그램 아이콘입니다.
- 76x76 픽셀 아이콘(선택 사항) - 저해상도 iPad의 응용 프로그램 아이콘입니다.
- 80x80 픽셀 아이콘 - 고해상도 iPhone/iPod/iPad의 내부 검색 아이콘입니다.
- 100x100 픽셀 아이콘 - Retina 디스플레이 iPad의 내부 검색 아이콘입니다.
- 114x114 픽셀 아이콘 - Retina 디스플레이 iPhone/iPod의 응용 프로그램 아이콘입니다.

- 120x120 픽셀 아이콘 - 고해상도 iPhone/iPod의 응용 프로그램 아이콘입니다.
- 152x152 픽셀 아이콘 - 고해상도 iPad의 응용 프로그램 아이콘입니다.
- 167x167 픽셀 아이콘 - 고해상도 iPad Pro의 응용 프로그램 아이콘입니다.
- 512x512 픽셀 아이콘 - 저해상도 iPhone/iPod/iPad의 응용 프로그램 아이콘입니다. iTunes에서 이 아이콘을 표시합니다. 512픽셀 PNG 파일은 개발 버전의 응용 프로그램을 테스트하는 데에만 사용됩니다. Apple App Store에 최종 응용 프로그램을 제출하는 경우 JPG 파일 형식의 512픽셀 이미지를 별도로 제출해야 합니다. IPA에 포함되지 않습니다.
- 1024x1024 픽셀 아이콘 - Retina 디스플레이 iPhone/iPod/iPad의 응용 프로그램 아이콘입니다.

iOS에서 자동으로 아이콘에 후광 효과가 추가되므로 소스 이미지에 효과를 적용할 필요는 없습니다. 이 기본 후광 효과를 제거하려면 응용 프로그램 설명자 파일의 InfoAdditions 요소에 다음을 추가하십시오.

```
<InfoAdditions>
                                    <![CDATA[
                                    <key>UIPrerenderedIcon</key>
                                    <true/>
                                    ]]>
                                    </InfoAdditions>
```

참고: iOS에서는 응용 프로그램 메타데이터가 응용 프로그램 아이콘에 ping 메타데이터로 삽입되므로 Abobe는 Apple iOS 앱 스토어에서 사용 가능한 AIR 응용 프로그램의 수를 추적할 수 있습니다. 이 아이콘 메타데이터로 인해 자신의 응용 프로그램이 AIR 응용 프로그램으로 식별되는 것을 원치 않는 경우에는 IPA 파일의 패키지화를 해제하고 아이콘 메타데이터를 제거한 후 다시 패키지화해야 합니다. 이 절차는 iOS에 대한 AIR 응용 프로그램 분석 기능 제외 문서에 설명되어 있습니다.

기타 도움말 항목

203페이지의 “icon”

204페이지의 “imageNxN”

Android 개발자: 아이콘 디자인 지침

iOS 휴먼 인터페이스 지침: 사용자 정의 아이콘 및 이미지 생성 지침

iOS 시작 이미지

응용 프로그램 아이콘 외에도 **Default.png**라는 시작 이미지를 하나 이상 제공해야 합니다. 서로 다른 방향, 해상도(고해상도 Retina 디스플레이 및 16:9 종횡비 포함) 및 장치에 대해 각각의 시작 이미지를 포함할 수도 있습니다. 응용 프로그램이 URL을 통해 호출될 때 사용할 다른 시작 이미지를 포함할 수도 있습니다.

시작 이미지 파일은 응용 프로그램 설명자에서 참조되지 않으며 루트 응용 프로그램 디렉토리에 넣어야 합니다. 파일을 하위 디렉토리에 넣지 마십시오.

파일 이름 지정 체계

다음 스킴에 따라 이미지의 이름을 지정하십시오.

```
basename + screen size modifier + urischeme + orientation + scale + device + .png
```

파일 이름의 **basename** 부분은 유일한 필수 부분입니다. 이 부분은 **Default**(대문자 D 사용)이거나 응용 프로그램 설명자의 InfoAdditions 요소에서 UILaunchImageFile 키를 사용하여 지정한 이름입니다.

화면 크기 수정자 부분은 표준 화면 크기 중 하나가 아닌 경우 화면의 크기를 지정합니다. 이 수정자는 iPhone 5 및 iPod Touch(5세대) 같은 16:9 종횡비 화면의 iPhone 및 iPod Touch 모델에만 적용됩니다. 이 수정자에 대해 지원되는 유일한 값은 -568h입니다. 이러한 장치에서는 고해상도(Retina) 디스플레이를 지원하므로 화면 크기 수정자는 항상 크기 조절 수정자가 @2x인 이미지와 함께 사용됩니다. 이러한 장치의 전체 기본 시작 이미지 이름은 Default-568h@2x.png입니다.

urischeme 부분은 URI 스킴을 식별하는 데 사용되는 문자열입니다. 이 부분은 응용 프로그램에서 하나 이상의 사용자 정의 URL 스킴을 지원하는 경우에만 적용됩니다. 예를 들어 example://foo와 같은 링크를 통해 응용 프로그램을 호출할 수 있는 경우 -example을 시작 이미지 파일 이름의 스킴 부분으로 사용하십시오.

orientation 부분은 응용 프로그램이 시작될 때 장치의 방향에 따라 사용할 여러 시작 이미지를 지정하는 데 활용됩니다. 이 부분은 iPad 응용 프로그램용 이미지에만 적용되며, 응용 프로그램이 시작될 때의 장치 방향을 나타내는 다음 값 중 하나일 수 있습니다.

- -Portrait
- -PortraitUpsideDown
- -Landscape
- -LandscapeLeft
- -LandscapeRight

scale 부분은 고해상도(Retina) 디스플레이에 사용되는 시작 이미지의 경우 @2x(iPhone 4, iPhone 5 및 iPhone 6용) 또는 @3x(iPhone 6 plus용)입니다. 표준 해상도 디스플레이에 사용되는 이미지에 대해서는 scale 부분을 생략하십시오. 또한 iPhone 5 및 iPod Touch(5세대) 같이 길이가 긴 장치용 시작 이미지의 경우에는 basename 부분 뒤와 다른 부분 앞에 화면 크기 수정자 -528h를 지정해야 합니다.

device 부분은 핸드헬드 장치 및 휴대폰용 시작 이미지를 지정하는 데 사용됩니다. 이 부분은 해당 응용 프로그램이 단일 응용 프로그램 바이너리를 통해 핸드헬드 장치와 타블렛을 모두 지원하는 범용 응용 프로그램인 경우에 사용됩니다. 사용 가능한 값은 ~ipad 또는 ~iphone이어야 합니다(iPhone 및 iPod Touch 모두).

iPhone의 경우 세로 종횡비 이미지만 포함할 수 있습니다. 그러나 iPhone 6 plus의 경우에는 가로 이미지도 추가할 수 있습니다. 표준 해상도 장치에는 320x480픽셀 이미지를, 고해상도 장치에는 640x960픽셀 이미지를, 그리고 iPhone 5 및 iPod Touch(5세대) 같은 16:9 종횡비 장치에는 640x1136픽셀 이미지를 각각 사용합니다.

iPad의 경우 다음 지침에 따라 이미지를 포함합니다.

- AIR 3.3 이하 - 비전체 화면 이미지: 가로(보통 해상도의 경우 1024x748, 고해상도의 경우 2048x1496) 및 세로(보통 해상도의 경우 768x1004, 고해상도의 경우 1536x2008) 종횡비 이미지를 모두 포함할 수 있습니다.
- AIR 3.4 이상 - 전체 화면 이미지: 가로(보통 해상도의 경우 1024x768, 고해상도의 경우 2048x1536) 및 세로(보통 해상도의 경우 768x1024, 고해상도의 경우 1536x2048) 종횡비 이미지를 모두 포함할 수 있습니다. 비전체 화면 응용 프로그램용으로 전체 화면 이미지를 패키지화할 경우, 상단의 20픽셀(고해상도인 경우 상단 40픽셀)은 상태 표시줄로 가려집니다. 이 영역에는 중요한 정보를 표시하지 않습니다.

예제

다음 표에는 가장 광범위한 장치 및 방향을 지원하며 example:// 스킴을 사용하는 URL로 시작될 수 있는 가상의 응용 프로그램에 포함할 수 있는 시작 이미지의 예가 나와 있습니다.

파일 이름	이미지 크기	사용
Default.png	320 x 480	iPhone, 표준 해상도
Default@2x.png	640 x 960	iPhone, 고해상도
Default-568h@2x.png	640 x 1136	iPhone, 고해상도, 16:9 종횡비
Default-Portrait.png	768 x 1004 (AIR 3.3 이하) 768 x 1024(AIR 3.4 이상)	iPad, 세로 방향
Default-Portrait@2x.png	1536 x 2008 (AIR 3.3 이하) 1536 x 2048(AIR 3.4 이상)	iPad, 고해상도, 세로 방향

파일 이름	이미지 크기	사용
Default-PortraitUpsideDown.png	768 x 1004 (AIR 3.3 이하) 768 x 1024(AIR 3.4 이상)	iPad, 위/아래가 뒤집힌 세로 방향
Default-PortraitUpsideDown@2x.png	1536 x 2008 (AIR 3.3 이하) 1536 x 2048(AIR 3.4 이상)	iPad, 고해상도, 뒤집힌 세로 방향
Default-Landscape.png	1024 x 768	iPad, 왼쪽 가로 방향
Default-LandscapeLeft@2x.png	2048 x 1536	iPad, 고해상도, 왼쪽 가로 방향
Default-LandscapeRight.png	1024 x 768	iPad, 오른쪽 가로 방향
Default-LandscapeRight@2x.png	2048 x 1536	iPad, 고해상도, 오른쪽 가로 방향
Default-example.png	320 x 480	표준 iPhone에서 example:// URL
Default-example@2x.png	640 x 960	고해상도 iPhone에서 example:// URL
Default-example~ipad.png	768 x 1004	세로 방향의 iPad에서 example:// URL
Default-example-Landscape.png	1024 x 768	가로 방향의 iPad에서 example:// URL

이 예제에서는 한 가지 접근 방법만 보여 줍니다. 예를 들어 Default.png 이미지를 iPad에 사용하고 Default~iphone.png 및 Default@2x~iphone.png를 iPhone 및 iPod에 대한 특정 시작 이미지로 지정할 수 있습니다.

참고 사항

iOS 응용 프로그래밍 안내서: 응용 프로그램 시작 이미지

iOS 장치용으로 패키지화할 시작 이미지

장치	해상도(픽셀)	시작 이미지 이름	방향
iPhone			
iPhone 4(Retina 아님)	640 x 960	Default~iphone.png	세로
iPhone 4, 4s	640 x 960	Default@2x~iphone.png	세로
iPhone 5, 5c, 5s	640 x 1136	Default-568h@2x~iphone.png	세로
iPhone6, iPhone7	750 x 1334	Default-375w-667h@2x~iphone.png	세로
iPhone6+, iPhone7+	1242 x 2208	Default-414w-736h@3x~iphone.png	세로
iPhone6+, iPhone7+	2208 x 1242	Default-Landscape-414w-736h@3x~iphone.png	가로
iPad			
iPad 1, 2	768 x 1024	Default-Portrait~ipad.png	세로
iPad 1, 2	768 x 1024	Default-PortraitUpsideDown~ipad.png	위/아래가 뒤집힌 세로
iPad 1, 2	1024 x 768	Default-Landscape~ipad.png	왼쪽 가로
iPad 1, 2	1024 x 768	Default-LandscapeRight~ipad.png	오른쪽 가로
iPad 3, Air	1536 x 2048	Default-Portrait@2x~ipad.png	세로

iPad 3, Air	1536 x 2048	Default-PortraitUpsideDown@2x~ipad.png	위/아래가 뒤집힌 세로
iPad 3, Air	2048 x 1536	Default-LandscapeLeft@2x~ipad.png	왼쪽 가로
iPad 3, Air	2048 x 1536	Default-LandscapeRight@2x~ipad.png	오른쪽 가로
iPad Pro	2048 x 2732	Default-Portrait@2x.png	세로
iPad Pro	2732 x 2048	Default-Landscape@2x.png	가로

아트 지침

크기만 올바르면 어떤 아트든 시작 이미지로 만들 수 있습니다. 그러나 일반적으로 이미지는 응용 프로그램의 초기 상태와 일치하는 것이 좋습니다. 응용 프로그램 시작 화면의 스크린 샷을 찍어서 이러한 시작 이미지를 만들 수 있습니다.

1 iOS 장치에서 응용 프로그램을 엽니다. 사용자 인터페이스의 첫 화면이 나타나면 화면 아래에 있는 [홈] 버튼을 누르고 있습니다. [홈] 버튼을 누른 상태에서 장치의 맨 위에 있는 [잠자기/깨우기]버튼을 누릅니다. 이렇게 하면 스크린 샷이 생성되어 카메라 롤로 보내집니다.

2 iPhoto 또는 기타 사진 전송 응용 프로그램에서 개발 컴퓨터로 이미지를 전송합니다.

응용 프로그램이 여러 가지 언어로 지역화되는 경우 시작 이미지에 텍스트를 포함하지 마십시오. 시작 이미지는 불변하므로 텍스트가 다른 언어와 맞지 않게 됩니다.

참고 사항

iOS 휴먼 인터페이스 지침: 시작 이미지

무시되는 설정

휴대 장치의 응용 프로그램은 기본 윈도우 또는 데스크톱 운영 체제 기능에 적용되는 응용 프로그램 설정을 무시합니다. 무시되는 설정은 다음과 같습니다.

- allowBrowserInvocation
- customUpdateUI
- fileTypes
- height
- installFolder
- maximizable
- maxSize
- minimizable
- minSize
- programMenuFolder
- resizable
- systemChrome
- title
- transparent
- visible
- width
- x

- y

모바일 AIR 응용 프로그램 패키지화

ADT -package 명령을 사용하여 휴대 장치용 AIR 응용 프로그램의 응용 프로그램 패키지를 만듭니다. -target 매개 변수는 패키지가 생성될 모바일 플랫폼을 지정합니다.

Android 패키지
Android의 AIR 응용 프로그램은 AIR 패키지 포맷 대신 Android 응용 프로그램 패키지 포맷(APK)을 사용합니다.

APK 대상 유형을 사용하여 ADT를 통해 제작된 패키지는 Android Market으로 전송할 수 있는 포맷으로 지정됩니다. Android Market에는 전송되는 응용 프로그램이 충족시켜야 하는 요구 사항이 있습니다. 최종 패키지를 만들기 전에 최신 요구 사항을 검토해야 합니다. Android 개발자: Market에서 제작을 참조하십시오.

iOS 응용 프로그램과 달리, Android 응용 프로그램은 일반 AIR 코드 서명 인증서를 사용하여 서명할 수 있습니다. 그러나 응용 프로그램을 Android Market에 전송하려면 인증서가 Market 규칙을 준수해야 합니다. 즉, 인증서가 적어도 2033년까지 유효해야 합니다. 그러한 인증서는 ADT -certificate 명령을 사용하여 만들 수 있습니다.

응용 프로그램이 Google 마켓으로부터 AIR 다운로드를 요구하는 것을 허용하지 않는 대체 마켓으로 응용 프로그램을 전송하려면 ADT의 -airDownloadURL 매개 변수를 사용하여 대체 다운로드 URL을 지정하면 됩니다. 필요한 버전의 AIR 런타임이 없는 사용자가 응용 프로그램을 시작하는 경우 지정된 URL로 이동합니다. 자세한 내용은 150페이지의 “ADT package 명령”을 참조하십시오.

기본적으로, ADT는 공유 런타임을 사용하여 Android 응용 프로그램을 패키징합니다. 따라서 해당 응용 프로그램을 실행하려면 사용자가 장치에 별도의 AIR 런타임을 설치해야 합니다.

참고: ADT가 전용 런타임을 사용하는 APK를 강제로 만들도록 하려면 target apk-captive-runtime을 사용하십시오.

iOS 패키지
iOS의 AIR 응용 프로그램은 기본 AIR 포맷 대신 iOS 패키지 포맷(IPA)을 사용합니다.

ipa-app-store 대상 유형 그리고 올바른 코드 서명 인증서와 프로비저닝 프로파일을 사용하여 ADT를 통해 제작된 패키지는 Apple App Store로 전송할 수 있는 포맷으로 되어 있습니다. 애드혹 배포를 위한 응용 프로그램을 패키지화하려면 **ipa-ad-hoc** 대상 유형을 사용하십시오.

Apple에서 발행한 올바른 개발자 인증서를 사용하여 응용 프로그램에 서명해야 합니다. 응용 프로그램 전송 이전의 최종 패키지화에 사용되는 테스트 빌드를 만들 때는 다른 인증서가 사용됩니다.

Ant를 사용하여 iOS 응용 프로그램을 패키지화하는 방법의 예를 확인하려면 Piotr Walczyszyn: ADT 명령 및 ANT 스크립트를 사용하여 iOS 디바이스용 AIR 응용 프로그램 패키지화를 참조하십시오.

ADT를 사용하여 패키지화

AIR SDK 버전 2.6 이상에서는 iOS와 Android에 대한 패키지화를 모두 지원합니다. 패키지화하기 전에 모든 ActionScript, MXML 및 확장 코드를 컴파일해야 합니다. 코드 서명 인증서도 있어야 합니다.

ADT 명령 및 옵션에 대한 자세한 참조는 149페이지의 “ADT(AIR Developer Tool)”를 참조하십시오.

Android APK 패키지

APK 패키지 만들기
APK 패키지를 만들려면 대상 유형을 릴리스 빌드에 대해 **apk**로, 디버그 빌드에 대해 **apk-debug**로 또는 릴리스 모드 빌드에 대해 **apk-emulator**로 설정하는 ADT package 명령을 사용하십시오.

```
adt -package
                                -target apk
                                -storetype pkcs12 -keystore ../codesign.p12
                                myApp.apk
                                myApp-app.xml
                                myApp.swf icons
```

한 행에 전체 명령을 입력합니다. 위 예제의 행 분리는 읽기 쉽게 표시하기 위한 것일 뿐입니다. 또한 이 예제에서는 ADT 도구의 경로가 명령줄 셸의 경로 정의에 있다고 가정합니다. 도움말은 277페이지의 "path 환경 변수"를 참조하십시오.

응용 프로그램 파일이 들어 있는 디렉토리에서 명령을 실행해야 합니다. 이 예제의 응용 프로그램 파일은 myApp-app.xml(응용 프로그램 설명자 파일), myApp.swf 및 icons 디렉토리입니다.

표시된 것과 같은 명령을 실행하면 ADT에서 키 저장소 암호를 묻습니다. 입력하는 암호 문자는 표시되지 않습니다. 암호 문자를 표시하려면 입력을 마치고 Enter 키를 누르면 됩니다.

참고: 기본적으로 모든 AIR Android 응용 프로그램의 패키지 이름에는 air. 접두어가 있습니다. 이 기본 비헤이비어를 옵트아웃하려면 해당 컴퓨터에서 환경 변수 AIR_NOANDROIDFLAIR를 **true**로 설정하십시오.

기본 확장을 사용하는 응용 프로그램용 APK 패키지 만들기

기본 확장을 사용하는 응용 프로그램에 대한 APK 패키지를 만들려면 일반적인 패키지 옵션 외에 -extdir 옵션을 추가합니다. 리소스/라이브러리를 공유하는 여러 ANE의 경우, 경고를 발생시키기 전에 ADT가 단일 리소스/라이브러리만 선택하고 다른 중복 항목은 무시합니다. 이 옵션은 응용 프로그램에 사용되는 ANE 파일이 포함된 디렉토리를 지정합니다. 예를 들면 다음과 같습니다.

```
adt -package
                                -target apk
                                -storetype pkcs12 -keystore ../codesign.p12
                                myApp.apk
                                myApp-app.xml
                                -extdir extensionsDir
                                myApp.swf icons
```

고유한 버전의 AIR 런타임이 포함된 APK 패키지 생성

응용 프로그램과 전용 버전의 AIR 런타임이 모두 포함된 APK 패키지를 만들려면 apk-captive-runtime 대상을 사용합니다. 이 옵션은 응용 프로그램에 사용되는 ANE 파일이 포함된 디렉토리를 지정합니다. 예를 들면 다음과 같습니다.

```
adt -package
                                -target apk-captive-runtime
                                -storetype pkcs12 -keystore ../codesign.p12
                                myApp.apk
                                myApp-app.xml
                                myApp.swf icons
```

이 방법 사용 시 발생할 수 있는 단점에는 다음이 포함됩니다.

- Adobe에서 보안 패치를 게시하는 경우 중요한 보안 픽스를 사용자가 자동으로 사용할 수 없습니다.
- 응용 프로그램의 RAM 공간이 커집니다.

참고: 런타임을 하나로 묶는 경우 ADT에서는 응용 프로그램에 INTERNET 및 BROADCAST_STICKY 권한을 추가합니다. 이러한 권한은 AIR 런타임에 필요한 권한입니다.

디버그 APK 패키지 만들기

디버거와 함께 사용할 수 있는 응용 프로그램 버전을 만들려면 apk-debug를 대상으로 사용하고 연결 옵션을 지정하십시오.

```
adt -package
                                -target apk-debug
                                -connect 192.168.43.45
                                -storetype pkcs12 -keystore ../codesign.p12
                                myApp.apk
                                myApp-app.xml
                                myApp.swf icons
```

-connect 플래그는 네트워크를 통해 원격 디버거에 연결할 장치의 AIR 런타임을 알려 줍니다. USB를 통해 디버깅하려면 디버그 연결에 사용할 TCP 포트를 지정하는 -listen 플래그를 대신 지정해야 합니다.

```
adt -package
                                -target apk-debug
                                -listen 7936
                                -storetype pkcs12 -keystore ../codesign.p12
                                myApp.apk
                                myApp-app.xml
                                myApp.swf icons
```

대부분의 디버깅 기능이 작동하도록 하려면 디버깅을 사용한 상태로 응용 프로그램 SWF 및 SWC도 컴파일해야 합니다. 164페이지의 “디버거 연결 옵션”에서 -connect 및 -listen 플래그에 대한 전체 설명을 참조하십시오.

참고: 기본적으로 ADT는 apk-debug 대상과 함께 응용 프로그램을 패키지화하는 동안 Android 응용 프로그램과 함께 AIR 런타임의 전용 복사본을 패키지화합니다. ADT가 외부 런타임을 사용하는 APK를 강제로 만들도록 하려면 AIR_ANDROID_SHARED_RUNTIME 환경 변수를 true로 설정하십시오.

Android에서는 응용 프로그램이 네트워크를 통해 디버거를 실행 중인 컴퓨터에 연결하려면 응용 프로그램도 인터넷에 대한 액세스 권한을 가지고 있어야 합니다. 70페이지의 “Android 권한”을 참조하십시오.

Android 에뮬레이터에서 사용할 APK 패키지 만들기

Android 에뮬레이터에서 디버그 APK 패키지를 사용할 수 있지만 릴리스 모드 패키지는 사용할 수 없습니다. 에뮬레이터에서 사용할 릴리스 모드 APK 패키지를 만들려면 ADT package 명령을 사용하여 대상 유형을 **apk-emulator**로 설정하십시오.

```
adt -package -target apk-emulator -storetype pkcs12 -keystore ../codesign.p12 myApp.apk myApp-app.xml
myApp.swf icons
```

이 예제에서는 ADT 도구의 경로가 명령줄 셸의 경로 정의에 있다고 가정합니다. 도움말은 277페이지의 “path 환경 변수”를 참조하십시오.

AIR 또는 AIRI 파일로부터 APK 패키지 만들기

기존 AIR 또는 AIRI 파일로부터 APK 패키지를 바로 만들 수 있습니다.

```
adt -target apk -storetype pkcs12 -keystore ../codesign.p12 myApp.apk myApp.air
```

AIR 파일은 응용 프로그램 설명자 파일에서 AIR 2.5 이상의 네임스페이스를 사용해야 합니다.

Android x86 플랫폼용 APK 패키지 만들기

AIR 14부터는 -arch 인수를 사용하여 Android x86 플랫폼용 APK를 패키지화할 수 있습니다. 예를 들면 다음과 같습니다.

```
adt     -package
                                -target apk-debug
                                -listen 7936
                                -arch x86
                                -storetype pkcs12 -keystore ../codesign.p12
                                myApp.apk
                                myApp-app.xml
                                myApp.swf icons
```

iOS 패키지

iOS에서 ADT는 SWF 파일 바이트 코드 및 기타 소스 파일을 기본 iOS 응용 프로그램으로 변환합니다.

1 Flash Builder, Flash Professional 또는 명령줄 컴파일러를 사용하여 SWF 파일을 만듭니다.

2 명령줄 셸 또는 터미널을 열고 iPhone 응용 프로그램의 프로젝트 폴더로 이동합니다.

3 그런 다음 ADT 도구에서 다음 구문을 사용하여 IPA 파일을 만듭니다.

```
adt -package
                                        -target [ipa-test | ipa-debug | ipa-app-store | ipa-ad-hoc |
                                        ipa-debug-interpreter | ipa-debug-interpreter-simulator
                                        ipa-test-interpreter | ipa-test-interpreter-simulator]
                                        -provisioning-profile PROFILE_PATH
                                        SIGNING_OPTIONS
                                        TARGET_IPA_FILE
                                        APP_DESCRIPTOR
                                        SOURCE_FILES
                                        -extdir extension-directory
                                        -platformsdk path-to-iossdk or path-to-ios-simulator-sdk
```

ADT 응용 프로그램의 전체 경로를 포함하도록 참조 adt를 변경합니다. ADT 응용 프로그램이 AIR SDK의 bin 하위 디렉토리에 설치됩니다.

만들려는 iPhone 응용 프로그램 유형에 해당하는 -target 옵션을 선택합니다.

- -target ipa-test - 응용 프로그램의 특정 버전을 개발자 iPhone에서 테스트용으로 빠르게 컴파일하려면 이 옵션을 선택합니다. 또한 ipa-test-interpreter를 사용하여 더욱 빠르게 컴파일하거나 ipa-test-interpreter-simulator를 사용하여 iOS 시뮬레이터에서 실행할 수 있습니다.
- -target ipa-debug - 응용 프로그램의 디버그 버전을 개발자 iPhone에서 테스트용으로 컴파일하려면 이 옵션을 선택합니다. 이 옵션을 선택하면 디버그 세션을 사용하여 iPhone 응용 프로그램에서 trace() 출력을 수신할 수 있습니다.

  다음 -connect 옵션(CONNECT_OPTIONS) 중 하나를 포함하여 디버거를 실행하는 개발 컴퓨터의 IP 주소를 지정할 수 있습니다.

  - -connect - 응용 프로그램에서 응용 프로그램을 컴파일하는 데 사용되는 개발 컴퓨터의 디버그 세션에 wifi를 통해 연결을 시도합니다.
  - -connect IP_ADDRESS - 응용 프로그램에서 지정된 IP 주소를 사용하는 컴퓨터의 디버그 세션에 wifi를 통해 연결을 시도합니다. 예를 들면 다음과 같습니다.

    ```
    -target ipa-debug -connect 192.0.32.10
    ```

  - -connect HOST_NAME - 응용 프로그램에서 지정된 호스트 이름을 사용하는 컴퓨터의 디버그 세션에 wifi를 통해 연결을 시도합니다. 예를 들면 다음과 같습니다.

    ```
    -target ipa-debug -connect bobroberts-mac.example.com
    ```

  -connect 옵션은 선택 사항입니다. 지정되지 않은 경우, 결과 디버그 응용 프로그램에서 호스팅되는 디버거에 연결하려고 하지 않습니다. 또는 96페이지의 “USB를 통해 FDB를 사용하여 원격 디버깅”에서 설명한 대로 -connect 대신 -listen을 지정하는 방식으로 USB 디버깅을 활성화할 수도 있습니다.

  디버그 연결이 실패하는 경우 디버깅 호스트 컴퓨터의 IP 주소를 입력하라는 메시지가 사용자에게 표시됩니다. 장치가 WiFi에 연결되어 있지 않은 경우 연결이 실패할 수 있습니다. 장치가 연결되어 있지만 디버깅 호스트 컴퓨터의 방화벽 내에 있지 않은 경우에도 이러한 문제가 발생할 수 있습니다.

  또한 ipa-debug-interpreter를 사용하여 더욱 빠르게 컴파일하거나 ipa-debug-interpreter-simulator를 사용하여 iOS 시뮬레이터에서 실행할 수 있습니다.

  자세한 내용은 91페이지의 “모바일 AIR 응용 프로그램 디버깅”을 참조하십시오.
- -target ipa-ad-hoc - 애드혹 배포용 응용 프로그램을 만들려면 이 옵션을 선택합니다. 자세한 내용은 Apple iPhone 개발자 센터를 참조하십시오.

- -target ipa-app-store - Apple App Store에 배포할 IPA 파일의 최종 버전을 만들려면 이 옵션을 선택합니다.

PROFILE_PATH를 응용 프로그램 프로비저닝 프로파일 파일의 경로로 바꿉니다. 프로비저닝 프로파일에 대한 자세한 내용은 60페이지의 "iOS 설정"을 참조하십시오.

응용 프로그램을 iOS 시뮬레이터에서 실행하기 위해 빌드하는 경우 -platformsdk 옵션을 사용하여 iOS 시뮬레이터 SDK를 가리킵니다.

iPhone 개발자 인증서와 암호를 참조하도록 SIGNING_OPTIONS를 바꿉니다. 다음 구문을 사용합니다.

```
-storetype pkcs12 -keystore P12_FILE_PATH -storepass PASSWORD
```

P12_FILE_PATH를 P12 인증서 파일의 경로로 바꿉니다. **PASSWORD**를 인증서 암호로 바꿉니다. 다음 예제를 참조하십시오. P12 인증서 파일에 대한 자세한 내용은 177페이지의 "개발자 인증서를 P12 키 저장소 파일로 변환"을 참조하십시오.

참고: iOS 시뮬레이터용으로 패키지화하는 경우 자체 서명된 인증서를 사용할 수 있습니다.

응용 프로그램 설명자 파일을 참조하도록 APP_DESCRIPTOR를 바꿉니다.

프로젝트의 기본 SWF 파일과 포함할 다른 에셋을 참조하도록 SOURCE_FILES를 바꿉니다. 사용자 정의 응용 프로그램 설명자 파일 또는 Flash Professional의 응용 프로그램 설정 대화 상자에서 정의한 모든 아이콘 파일의 경로를 포함합니다. 또한 초기 화면 아트 파일 Default.png도 추가합니다.

-extdir ***extension-directory*** 옵션을 사용하여 응용 프로그램에 사용되는 ANE 파일(기본 확장)이 포함된 디렉토리를 지정합니다. 응용 프로그램에 기본 확장이 사용되지 않는 경우에는 이 옵션을 포함하지 마십시오.

중요: Resources라는 응용 프로그램 디렉토리에 하위 디렉토리를 만들어서는 안 됩니다. 런타임에서 해당 이름이 IPA 패키지 구조를 따르는 폴더를 자동으로 만듭니다. Resources 폴더를 직접 만들면 심각한 충돌이 발생합니다.

디버깅을 위한 iOS 패키지 만들기

테스트 장치에 설치할 iOS 패키지를 만들려면 ADT package 명령을 사용하여 대상 유형을 **ios-debug**로 설정하십시오. 이 명령을 실행하기 전에 미리 Apple로부터 개발 코드 서명 인증서와 프로비저닝 프로파일을 얻어야 합니다.

```
adt -package
                                -target ipa-debug
                                -storetype pkcs12 -keystore ../AppleDevelopment.p12
                                -provisioning-profile AppleDevelopment.mobileprofile
                                -connect 192.168.0.12 | -listen
                                myApp.ipa
                                myApp-app.xml
                                myApp.swf icons Default.png
```

참고: ipa-debug-interpreter를 사용하여 더욱 빠르게 컴파일하거나 ipa-debug-interpreter-simulator를 사용하여 iOS 시뮬레이터에서 실행할 수도 있습니다.

한 행에 전체 명령을 입력합니다. 위 예제의 행 분리는 읽기 쉽게 표시하기 위한 것일 뿐입니다. 또한 이 예제에서는 ADT 도구의 경로가 명령줄 셸의 경로 정의에 있다고 가정합니다. 도움말은 277페이지의 "path 환경 변수"를 참조하십시오.

응용 프로그램 파일이 들어 있는 디렉토리에서 명령을 실행해야 합니다. 이 예제의 응용 프로그램 파일은 myApp-app.xml(응용 프로그램 설명자 파일), myApp.swf, icons 디렉토리 및 Default.png 파일입니다.

Apple에서 발행한 올바른 배포 인증서를 사용하여 응용 프로그램에 서명해야 합니다. 다른 코드 서명 인증서는 사용할 수 없습니다.

wifi 디버깅을 위해 -connect 옵션을 사용합니다. 응용 프로그램에서 지정된 IP 또는 호스트 이름에서 실행되는 Flash Debugger(FDB)를 사용하여 디버그 세션을 시작하려고 합니다. USB 디버깅을 위해 -listen 옵션을 사용합니다. 먼저 응용 프로그램을 시작한 후 FDB를 시작합니다. 그러면 실행 중인 응용 프로그램에 대해 디버그 세션이 시작됩니다. 자세한 내용은 94페이지의 "Flash 디버거에 연결"을 참조하십시오.

Apple App Store로 전송할 iOS 패키지 만들기

Apple App Store로 전송할 iOS 패키지를 만들려면 ADT package 명령을 사용하여 대상 유형을 **ios-app-store**로 설정하십시오. 이 명령을 실행하기 전에 미리 Apple로부터 배포 코드 서명 인증서와 프로비저닝 프로파일을 얻어야 합니다.

```
adt -package
                                -target ipa-app-store
                                -storetype pkcs12 -keystore ../AppleDistribution.p12
                                -provisioning-profile AppleDistribution.mobileprofile
                                myApp.ipa
                                myApp-app.xml
                                myApp.swf icons Default.png
```

한 행에 전체 명령을 입력합니다. 위 예제의 행 분리는 읽기 쉽게 표시하기 위한 것일 뿐입니다. 또한 이 예제에서는 ADT 도구의 경로가 명령줄 셸의 경로 정의에 있다고 가정합니다. 도움말은 277페이지의 “path 환경 변수”를 참조하십시오.

응용 프로그램 파일이 들어 있는 디렉토리에서 명령을 실행해야 합니다. 이 예제의 응용 프로그램 파일은 myApp-app.xml(응용 프로그램 설명자 파일), myApp.swf, icons 디렉토리 및 Default.png 파일입니다.

Apple에서 발행한 올바른 배포 인증서를 사용하여 응용 프로그램에 서명해야 합니다. 다른 코드 서명 인증서는 사용할 수 없습니다.

중요: Apple Strore에 응용 프로그램을 업로드하려면 Apple **Application Loader** 프로그램을 사용해야 합니다. Apple은 Mac OS X용 **Application Loader**만 제작합니다. 따라서 Windows 컴퓨터를 사용하여 iPhone용 AIR 응용 프로그램을 개발할 수는 있지만 해당 응용 프로그램을 App Store로 전송하려면 OS X(버전 10.5.3 이상)를 실행하는 컴퓨터에 액세스할 수 있어야 합니다. Application Loader 프로그램은 Apple iOS Developer Center에서 얻을 수 있습니다.

애드혹 배포를 위한 iOS 패키지 만들기

애드혹 배포를 위한 iOS 패키지를 만들려면 ADT package 명령을 사용하여 대상 유형을 **ios-ad-hoc**로 설정하십시오. 이 명령을 실행하기 전에 미리 Apple로부터 적절한 애드혹 배포 코드 서명 인증서와 프로비저닝 프로파일을 얻어야 합니다.

```
adt -package
                                -target ipa-ad-hoc
                                -storetype pkcs12 -keystore ../AppleDistribution.p12
                                -provisioning-profile AppleDistribution.mobileprofile
                                myApp.ipa
                                myApp-app.xml
                                myApp.swf icons Default.png
```

한 행에 전체 명령을 입력합니다. 위 예제의 행 분리는 읽기 쉽게 표시하기 위한 것일 뿐입니다. 또한 이 예제에서는 ADT 도구의 경로가 명령줄 셸의 경로 정의에 있다고 가정합니다. 도움말은 277페이지의 “path 환경 변수”를 참조하십시오.

응용 프로그램 파일이 들어 있는 디렉토리에서 명령을 실행해야 합니다. 이 예제의 응용 프로그램 파일은 myApp-app.xml(응용 프로그램 설명자 파일), myApp.swf, icons 디렉토리 및 Default.png 파일입니다.

Apple에서 발행한 올바른 배포 인증서를 사용하여 응용 프로그램에 서명해야 합니다. 다른 코드 서명 인증서는 사용할 수 없습니다.

기본 확장을 사용하는 응용 프로그램용 iOS 패키지 만들기

기본 확장을 사용하는 응용 프로그램용 iOS 패키지를 만들려면 -extdir 옵션과 함께 ADT 패키지 명령을 사용합니다. ADT 명령을 대상에 적절하게 사용합니다(ipa-app-store, ipa-debug, ipa-ad-hoc, ipa-test). 예를 들면 다음과 같습니다.

```
adt -package
                                -target ipa-ad-hoc
                                -storetype pkcs12 -keystore ../AppleDistribution.p12
                                -provisioning-profile AppleDistribution.mobileprofile
                                myApp.ipa
                                myApp-app.xml
                                -extdir extensionsDir
                                myApp.swf icons Default.png
```

한 행에 전체 명령을 입력합니다. 위 예제의 행 분리는 읽기 쉽게 표시하기 위한 것일 뿐입니다.

기본 확장과 관련하여 예제에서는 extensionsDir이라는 디렉토리가 명령이 실행되는 디렉토리에 있다고 가정합니다. extensionsDir 디렉토리에는 응용 프로그램에 사용되는 ANE 파일이 포함되어 있습니다.

모바일 AIR 응용 프로그램 디버깅

여러 가지 방법으로 모바일 AIR 응용 프로그램을 디버깅할 수 있습니다. 응용 프로그램 논리 문제를 발견하는 가장 간단한 방법은 ADL 또는 iOS 시뮬레이터를 사용하는 개발 컴퓨터에서 디버깅하는 것입니다. 장치에 응용 프로그램을 설치하고 데스크톱 컴퓨터에서 실행 중인 Flash 디버거를 사용하여 원격으로 디버깅할 수도 있습니다.

ADL을 사용하는 장치 시뮬레이션

대부분의 모바일 응용 프로그램 기능을 가장 빠르고 쉽게 테스트하고 디버깅하는 방법은 ADL(Adobe Debug Launcher) 유틸리티를 사용하는 개발 컴퓨터에서 응용 프로그램을 실행하는 것입니다. ADL은 응용 프로그램 설명자에서 supportedProfiles 요소를 사용하여 사용할 프로파일을 결정합니다. 목록에 여러 개의 프로파일이 있는 경우 ADL은 목록에서 첫 번째 프로파일을 사용합니다. ADL의 -profile 매개 변수를 사용하여 supportedProfiles 목록에서 다른 프로파일 중 하나를 선택할 수도 있습니다. 응용 프로그램 설명자에 supportedProfiles 요소를 포함하지 않는 경우에는 -profile 인수에 대해 아무 프로파일이나 지정하면 됩니다. 예를 들어 다음 명령을 사용하여 응용 프로그램을 시작함으로써 휴대 장치 프로파일을 시뮬레이트하십시오.

```
adl -profile mobileDevice myApp-app.xml
```

이와 같이 데스크톱에서 모바일 프로파일을 시뮬레이트할 때는 대상 휴대 장치와 좀 더 비슷한 환경에서 응용 프로그램이 실행됩니다. 모바일 프로파일에 속하지 않은 ActionScript API는 사용할 수 없습니다. 하지만 ADL은 서로 다른 휴대 장치의 기능을 구분하지 않습니다. 예를 들어 실제 대상 장치가 소프트 키를 사용하지 않는 경우에도 시뮬레이트된 소프트 키 누르기를 응용 프로그램에 보낼 수 있습니다.

ADL은 메뉴 명령을 통해 장치 방향 변경 및 소프트 키 입력의 시뮬레이션을 지원합니다. 휴대 장치 프로파일에서 ADL을 실행하면 ADL에서 장치 회전 또는 소프트 키 입력을 입력할 수 있는 메뉴를 표시하며, 이러한 메뉴는 응용 프로그램 윈도우 또는 데스크톱 메뉴 모음에 표시됩니다.

소프트 키 입력

ADL은 휴대 장치의 [뒤로], [메뉴] 및 [검색] 버튼에 대해 소프트 키 버튼을 시뮬레이트합니다. 모바일 프로파일을 사용하여 ADL을 실행할 때 표시되는 메뉴를 통해 시뮬레이트된 장치로 소프트 키를 보낼 수 있습니다.

장치 회전

ADL에서는 모바일 프로파일을 사용하여 ADL을 실행할 때 표시되는 메뉴를 통해 장치 회전을 시뮬레이트할 수 있습니다. 시뮬레이트된 장치는 오른쪽 또는 왼쪽으로 회전할 수 있습니다.

회전 시뮬레이션은 자동 방향을 사용하는 응용 프로그램에만 영향을 줍니다. 응용 프로그램 설명자에서 autoOrients 요소를 true로 설정하여 이 기능을 사용할 수 있습니다.

스크린 크기

ADL -screensize 매개 변수를 설정하여 다양한 크기의 스크린에서 응용 프로그램을 테스트할 수 있습니다. 미리 정의된 스크린 유형 중 하나에 대한 코드를 전달하거나, 일반 스크린 및 최대화된 스크린의 픽셀 크기를 나타내는 네 개의 값이 포함된 문자열을 전달할 수 있습니다.

항상 세로 레이아웃에 대한 픽셀 크기를 지정합니다. 이는 높이 값보다 작은 값을 폭에 지정해야 한다는 것을 의미합니다. 예를 들어 다음 명령은 ADL을 열어 Motorola Droid에 사용되는 스크린을 시뮬레이트합니다.

```
adl -screensize 480x816:480x854 myApp-app.xml
```

미리 정의된 스크린 유형의 목록은 144페이지의 “ADL 사용”을 참조하십시오.

제한

데스크톱 프로파일에서 지원되지 않는 일부 API는 ADL에서 시뮬레이트할 수 없습니다. 시뮬레이트되지 않는 API 중 일부는 다음과 같습니다.

- Accelerometer
- cacheAsBitmapMatrix
- CameraRoll
- CameraUI
- Geolocation
- 이러한 기능을 지원하지 않는 데스크톱 운영 체제에서의 멀티터치 및 동작
- SystemIdleMode

응용 프로그램에서 이러한 클래스를 사용하는 경우 실제 장치 또는 에뮬레이터에서 기능을 테스트해야 합니다.

마찬가지로 데스크톱의 ADL에서 실행될 때는 작동하지만 특정 유형의 휴대 장치에서는 작동하지 않는 API가 있습니다. 그 중 일부는 다음과 같습니다.

- Speex 및 AAC 오디오 코덱
- 접근성 및 화면 읽기 지원
- RTMPE
- ActionScript 바이트코드를 포함하는 SWF 파일 로드
- PixelBender 셰이더

ADL은 실행 환경을 완전히 복제하지 않기 때문에 응용 프로그램이 대상 장치에서 해당 기능을 사용하는지 여부를 테스트해야 합니다.

iOS 시뮬레이터를 사용한 장치 시뮬레이션

iOS 시뮬레이터(Mac 전용)를 사용하면 iOS 응용 프로그램을 빠르게 실행하고 디버깅할 수 있습니다. iOS 시뮬레이터를 사용하여 테스트하는 경우 개발자 인증서나 프로비저닝 프로파일이 필요하지 않습니다. 그러나 p12 인증서는 만들어야 합니다(자체 서명 가능).

기본적으로 ADT는 항상 iPhone 시뮬레이터를 실행합니다. 시뮬레이터 장치를 변경하려면 다음을 수행합니다.

- 아래의 명령을 사용하여 사용 가능한 시뮬레이터를 확인합니다.

  ```
  xcrun simctl list devices
  ```

  아래와 같이 결과가 표시됩니다.

  ```
  == Devices ==
  -iOS 10.0 -
  iPhone 5 (F6378129-A67E-41EA-AAF9-D99810F6BCE8) (Shutdown)
  iPhone 5s (5F640166-4110-4F6B-AC18-47BC61A47749) (Shutdown)
  iPhone 6 (E2ED9D38-C73E-4FF2-A7DD-70C55A021000) (Shutdown)
  iPhone 6 Plus (B4DE58C7-80EB-4454-909A-C38C4106C01B) (Shutdown)
  iPhone 6s (9662CB8A-2E88-403E-AE50-01FB49E4662B) (Shutdown)
  iPhone 6s Plus (BED503F3-E70C-47E1-BE1C-A2B7F6B7B63E) (Shutdown)
  iPhone 7 (71880D88-74C5-4637-AC58-1F9DB43BA471) (Shutdown)
  iPhone 7 Plus (2F411EA1-EE8B-486B-B495-EFC421E0A494) (Shutdown)
  iPhone SE (DF52B451-ACA2-47FD-84D9-292707F9F0E3) (Shutdown)
  iPad Retina (C4EF8741-3982-481F-87D4-700ACD0DA6E1) (Shutdown)
  ....
  ```

- 다음과 같이 환경 변수 AIR_IOS_SIMULATOR_DEVICE를 설정하여 특정 시뮬레이터를 선택할 수 있습니다.

```
export AIR_IOS_SIMULATOR_DEVICE = 'iPad Retina'
```

환경 변수를 설정한 후 프로세스를 다시 시작하고 원하는 시뮬레이터 장치에서 응용 프로그램을 실행합니다.

참고: iOS 시뮬레이터와 함께 ADT를 사용하는 경우 항상 -platformsdk 옵션을 포함하여 iOS 시뮬레이터 SDK에 대한 경로를 지정해야 합니다.

iOS 시뮬레이터에서 응용 프로그램을 실행하려면

1 다음 예제와 같이 -target ipa-test-interpreter-simulator 또는 -target ipa-debug-interpreter-simulator와 함께 adt -package 명령을 사용합니다.

```
adt -package
                                -target ipa-test-interpreter-simulator
                                -storetype pkcs12 -keystore Certificates.p12
                                -storepass password
                                myApp.ipa
                                myApp-app.xml
                                myApp.swf
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
```

참고: 이제 시뮬레이터의 경우 서명 옵션이 더 이상 필요하지 않습니다. 따라서 ADT에서 인정되지 않으므로 -keystore 플래그에 임의의 값을 제공할 수 있습니다.

2 다음 예제와 같이 adt -installApp 명령을 사용하여 iOS 시뮬레이터에서 응용 프로그램을 설치합니다.

```
adt -installApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -package sample_ipa_name.ipa
```

3 다음 예제와 같이 adt -launchApp 명령을 사용하여 iOS 시뮬레이터에서 응용 프로그램을 실행합니다.

참고: 기본적으로 adt -launchApp 명령은 iPhone 시뮬레이터에서 응용 프로그램을 실행합니다. iPad 시뮬레이터에서 응용 프로그램을 실행하려면 환경 변수 AIR_IOS_SIMULATOR_DEVICE = "iPad"를 내보낸 후 명령 adt -launchApp을 사용합니다.

```
adt -launchApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -appid sample_ipa_name
```

iOS 시뮬레이터에서 기본 확장을 테스트하려면 다음 extension.xml 예제와 같이 extension.xml 파일에서 iPhone-x86 플랫폼 이름을 사용하고 library.a(정적 라이브러리)(nativeLibrary 요소)를 지정합니다.

```
<extension xmlns="http://ns.adobe.com/air/extension/3.1">
                          <id>com.cnative.extensions</id>
                          <versionNumber>1</versionNumber>
                          <platforms>
                            <platform name="iPhone-x86">
                              <applicationDeployment>
                                <nativeLibrary>library.a</nativeLibrary>
                                <initializer>TestNativeExtensionsInitializer </initializer>
                                <finalizer>TestNativeExtensionsFinalizer </finalizer>
                              </applicationDeployment>
                            </platform>
                          </platforms>
                        </extension>
```

참고: iOS 시뮬레이터에서 기본 확장을 테스트하는 경우 장치용으로 컴파일된 정적 라이브러리(.a 파일)를 사용하지 마십시오. 그 대신에 시뮬레이터용으로 컴파일된 정적 라이브러리를 사용해야 합니다.

trace 문

데스크톱에서 모바일 응용 프로그램을 실행하면 ADL을 실행하는 데 사용되는 디버거 또는 터미널 윈도우에 trace 출력이 인쇄됩니다. 장치 또는 에뮬레이터에서 응용 프로그램을 실행할 때는 trace 출력을 보기 위한 원격 디버깅 세션을 설정할 수 있습니다. 지원되는 경우 장치 또는 운영 체제 제조업체에서 제공하는 소프트웨어 개발 도구를 사용하여 trace 출력을 볼 수도 있습니다.

어떠한 경우든 런타임이 trace 문을 출력하려면 응용 프로그램에 있는 SWF 파일이 디버깅이 사용되는 상태로 컴파일되어야 합니다.

Android의 원격 trace 문

Android 장치 또는 에뮬레이터에서 실행할 때는 Android SDK에 포함된 ADB(Android Debug Bridge) 유틸리티를 사용하여 Android 시스템 로그에서 trace 문 출력을 볼 수 있습니다. 응용 프로그램의 출력을 보려면 개발 컴퓨터의 명령 프롬프트 또는 터미널 윈도우에서 다음 명령을 실행하십시오.

```
tools/adb logcat air.MyApp:I *:S
```

여기서 **MyApp**은 응용 프로그램의 AIR 응용 프로그램 ID입니다. 인수 *:S는 다른 모든 프로세스에서 출력을 표시하지 않습니다. trace 출력 외에도 응용 프로그램에 대한 시스템 정보를 보려면 logcat 필터 사양에 ActivityManager를 포함할 수 있습니다.

```
tools/adb logcat air.MyApp:I ActivityManager:I *:S
```

이러한 명령 예제는 Android SDK 폴더에서 ADB를 실행 중이거나 Path 환경 변수에 SDK 폴더를 추가한 것으로 가정합니다.

참고: AIR 2.6+에서 ADB 유틸리티는 AIR SDK에 포함되어 있으며 lib/android/bin 폴더에서 찾을 수 있습니다.

iOS의 원격 trace 문

iOS 장치에서 실행되는 응용 프로그램에서 trace 문의 출력을 보려면 FDB(Flash Debugger)를 사용하여 원격 디버깅 세션을 설정해야 합니다.

기타 도움말 항목

Android Debug Bridge: logcat 로깅 사용

277페이지의 “path 환경 변수”

Flash 디버거에 연결

휴대 장치에서 실행 중인 응용 프로그램을 디버깅하려면 개발 컴퓨터에서 Flash 디버거를 실행하고 네트워크를 통해 연결하면 됩니다. 원격 디버깅을 사용하려면 다음을 수행해야 합니다.

- Android에서는 응용 프로그램 설명자에서 android:permission.INTERNET 권한을 지정합니다.
- 디버깅을 사용하는 상태로 응용 프로그램 SWF를 컴파일합니다.
- -target apk-debug(Android) 또는 -target ipa-debug(iOS)와 -connect(wifi 디버깅) 또는 -listen(USB 디버깅) 플래그를 사용하여 응용 프로그램을 패키지화합니다.

wifi를 통한 원격 디버깅의 경우, 장치에서 IP 주소 또는 정규화된 도메인 이름으로 Flash Debugger를 실행하는 컴퓨터의 TCP 포트 7935에 액세스할 수 있어야 합니다. USB를 통한 원격 디버깅의 경우, 장치에서 TCP 포트 7936 또는 -listen 플래그에 지정된 포트에 액세스할 수 있어야 합니다.

iOS의 경우 -target ipa-debug-interpreter 또는 -target ipa-debug-interpreter-simulator를 지정할 수도 있습니다.

Flash Professional을 사용한 원격 디버깅

응용 프로그램이 디버깅할 준비가 되었고 응용 프로그램 설명자에 권한이 설정되어 있으면 다음을 수행하십시오.

1 [AIR Android 설정] 대화 상자를 엽니다.

2 [배포] 탭 아래에서 다음을 수행합니다.

- 배포 유형으로 "장치 디버깅"을 선택합니다.
- [제작 후]에 대해 "연결된 Android 장치에 응용 프로그램 설치"를 선택합니다.
- [제작 후]에 대해 "연결된 Android 장치에서 응용 프로그램 실행"의 선택을 취소합니다.
- 필요한 경우 Android SDK의 경로를 설정합니다.

3 [제작]을 클릭합니다.

장치에서 응용 프로그램이 설치 및 실행됩니다.

4 [AIR Android 설정] 대화 상자를 닫습니다.

5 Flash Professional 메뉴에서 [디버그] > [원격 디버그 세션 시작] > ActionScript 3를 선택합니다.

Flash Professional의 [출력] 패널에 "플레이어의 연결을 기다리는 중"이라고 표시됩니다.

6 장치에서 응용 프로그램을 실행합니다.

7 Adobe AIR 연결 대화 상자에서 Flash 디버거를 실행하는 컴퓨터의 IP 주소 또는 호스트 이름을 입력한 후에 [확인]을 클릭합니다.

네트워크 연결을 통해 FDB를 사용하여 원격 디버깅

명령줄 FDB(Flash Debugger)를 사용하여 장치에서 실행 중인 응용 프로그램을 디버깅하려면 먼저 개발 컴퓨터에서 디버거를 실행한 후에 장치에서 응용 프로그램을 시작하십시오. 다음 절차에서는 AMXMLC, FDB 및 ADT 도구를 사용하여 장치에서 응용 프로그램을 컴파일, 패키지화 및 디버깅합니다. 이 예제에서는 결합된 Flex 및 AIR SDK를 사용하고 있으며 Path 환경 변수에 bin 디렉토리가 포함되어 있다고 가정합니다. 이러한 가정은 명령 예제를 단순화하기 위한 것일 뿐입니다.

1 터미널 또는 명령 프롬프트 윈도우를 열고 응용 프로그램의 소스 코드가 들어 있는 디렉토리로 이동합니다.

2 amxmlc로 응용 프로그램을 컴파일하여 디버깅이 사용되도록 합니다.

```
amxmlc -debug DebugExample.as
```

3 apk-debug 또는 ipa-debug 대상을 사용하여 응용 프로그램을 패키지화합니다.

```
Android
                                    adt -package -target apk-debug -connect -storetype pkcs12 -keystore
../../AndroidCert.p12 DebugExample.apk DebugExample-app.xml DebugExample.swf
                                    iOS
                                    adt -package -target ipa-debug -connect -storetype pkcs12 -keystore
../../AppleDeveloperCert.p12 -provisioning-profile test.mobileprovision DebugExample.apk DebugExample-
app.xml DebugExample.swf
```

디버깅에 항상 같은 호스트 이름 또는 IP 주소를 사용하는 경우에는 -connect 플래그 뒤에 해당 값을 넣을 수 있습니다. 그러면 응용 프로그램이 해당 IP 주소 또는 호스트 이름에 자동으로 연결하려고 시도합니다. 그렇지 않으면 디버깅을 시작할 때마다 장치에 대한 정보를 입력해야 합니다.

4 응용 프로그램을 설치합니다.

Android에서는 ADT -installApp 명령을 사용할 수 있습니다.

```
adt -installApp -platform android -package DebugExample.apk
```

iOS에서는 ADT -installApp 명령 또는 iTunes를 사용하여 응용 프로그램을 설치할 수 있습니다.

5 두 번째 터미널 또는 명령 윈도우에서 FDB를 실행합니다.

```
fdb
```

6 FDB 윈도우에서 run 명령을 입력합니다.

```
Adobe fdb (Flash Player Debugger) [build 14159]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
                                    Waiting for Player to connect
```

7 장치에서 응용 프로그램을 실행합니다.

8 장치 또는 에뮬레이터에서 응용 프로그램이 실행되면 Adobe AIR 연결 대화 상자가 열립니다. 응용 프로그램을 패키지화할 때 -connect 옵션을 사용하여 호스트 이름 또는 IP 주소를 지정한 경우에는 해당 주소를 사용하여 자동으로 연결하려고 시도합니다. 적절한 주소를 입력하고 [확인]을 누릅니다.

이 모드에서 디버거에 연결하려면 장치가 주소 또는 호스트 이름을 확인하고 TCP 포트 7935에 연결할 수 있어야 합니다. 이를 위해 네트워크 연결이 필요합니다.

9 원격 런타임이 디버거에 연결할 때는 FDB break 명령을 사용하여 중단점을 설정한 후에 continue 명령을 사용하여 실행을 시작할 수 있습니다.

```
(fdb) run
                                    Waiting for Player to connect
                                    Player connected; session starting.
                                    Set breakpoints and then type 'continue' to resume the session.
                                    [SWF]
Users:juser:Documents:FlashProjects:DebugExample:DebugExample.swf - 32,235 bytes after decompression
                                    (fdb) break clickHandler
                                    Breakpoint 1 at 0x5993: file DebugExample.as, line 14
                                    (fdb) continue
```

USB를 통해 FDB를 사용하여 원격 디버깅

AIR 2.6(Android) AIR 3.3(iOS)

USB 연결을 통해 응용 프로그램을 디버깅하려면 -listen 옵션(-connect 옵션 아님)을 사용하여 응용 프로그램을 패키지화합니다. -listen 옵션을 지정하면 응용 프로그램을 시작할 때 런타임이 TCP 포트 7936에서 Flash Debugger(FDB)로부터 연결을 수신 대기합니다. 그 후 -p 옵션을 사용하여 FDB를 실행하면 FDB가 연결을 시작합니다.

Android의 USB 디버깅 절차

데스크톱 컴퓨터에서 실행되는 Flash Debugger가 장치 또는 에뮬레이터에서 실행되는 AIR 런타임에 연결할 수 있도록 하려면 Android Debug Bridge(ADB - Android SDK의 유틸리티) 또는 iOS Debug Bridge(IDB - AIR SDK의 유틸리티)를 사용하여 장치 포트를 데스크톱 포트로 전달해야 합니다.

1 터미널 또는 명령 프롬프트 윈도우를 열고 응용 프로그램의 소스 코드가 들어 있는 디렉토리로 이동합니다.

2 amxmlc로 응용 프로그램을 컴파일하여 디버깅이 사용되도록 합니다.

```
amxmlc -debug DebugExample.as
```

3 적절한 디버그 대상(예: apk-debug)을 사용하여 응용 프로그램을 패키지화하고 -listen 옵션을 지정합니다.

```
adt -package -target apk-debug -listen -storetype pkcs12 -keystore ../../AndroidCert.p12 DebugExample.apk
DebugExample-app.xml DebugExample.swf
```

4 USB 케이블을 사용하여 장치를 디버그 컴퓨터에 연결합니다. 이 절차를 사용하여 에뮬레이터에서 실행 중인 응용 프로그램을 디버깅할 수도 있습니다. 이 경우에는 USB 연결이 필요하지 않거나 불가능합니다.

5 응용 프로그램을 설치합니다.

ADT -installApp 명령을 사용할 수 있습니다.

```
adt -installApp -platform android -package DebugExample.apk
```

6 Android ADB 유틸리티를 사용하여 장치 또는 에뮬레이터에서 데스크톱 컴퓨터로 TCP 포트 7936을 전달합니다.

```
adb forward tcp:7936 tcp:7936
```

7 장치에서 응용 프로그램을 실행합니다.

8 터미널 또는 명령 윈도우에서 -p 옵션을 사용하여 FDB를 실행합니다.

```
fdb -p 7936
```

9 FDB 윈도우에서 run 명령을 입력합니다.

```
Adobe fdb (Flash Player Debugger) [build 14159]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
```

10 FDB 유틸리티가 응용 프로그램에 연결하려고 시도합니다.

11 원격 연결이 설정되면 FDB break 명령을 사용하여 중단점을 설정한 후에 continue 명령을 사용하여 실행을 시작할 수 있습니다.

```
(fdb) run
                                    Player connected; session starting.
                                    Set breakpoints and then type 'continue' to resume the session.
                                    [SWF]
Users:juser:Documents:FlashProjects:DebugExample:DebugExample.swf - 32,235 bytes after decompression
                                    (fdb) break clickHandler
                                    Breakpoint 1 at 0x5993: file DebugExample.as, line 14
                                    (fdb) continue
```

참고: 포트 번호 7936은 AIR 런타임과 FDB 모두에서 USB 디버깅의 기본값으로 사용됩니다. ADT -listen 포트 매개 변수 및 FDB -p 포트 매개 변수를 통해 사용할 다른 포트를 지정할 수 있습니다. 이 경우에는 Android Debug Bridge 유틸리티를 사용하여 ADT에 지정된 포트 번호를 FDB에 지정된 포트로 전달해야 합니다(adb forward tcp:adt_listen_port# tcp:fdb_port#).

iOS의 USB 디버깅 절차

데스크톱 컴퓨터에서 실행되는 Flash Debugger가 장치 또는 에뮬레이터에서 실행되는 AIR 런타임에 연결할 수 있도록 하려면 iOS Debug Bridge(IDB - AIR SDK의 유틸리티)를 사용하여 장치 포트를 데스크톱 포트로 전달해야 합니다.

1 터미널 또는 명령 프롬프트 윈도우를 열고 응용 프로그램의 소스 코드가 들어 있는 디렉토리로 이동합니다.

2 amxmlc로 응용 프로그램을 컴파일하여 디버깅이 사용되도록 합니다.

```
amxmlc -debug DebugExample.as
```

3 적절한 디버그 대상(예: ipa-debug 또는 ipa-debug-interpreter)을 사용하여 응용 프로그램을 패키지화하고 -listen 옵션을 지정합니다.

```
adt -package -target ipa-debug-interpreter -listen 16000
                                    xyz.mobileprovision -storetype pkcs12 -keystore Certificates.p12
                                    -storepass pass123 OutputFile.ipa InputFile-app.xml InputFile.swf
```

4 USB 케이블을 사용하여 장치를 디버그 컴퓨터에 연결합니다. 이 절차를 사용하여 에뮬레이터에서 실행 중인 응용 프로그램을 디버깅할 수도 있습니다. 이 경우에는 USB 연결이 필요하지 않거나 불가능합니다.

5 iOS 장치에서 응용 프로그램을 설치하고 시작합니다. AIR 3.4 이상에서는 adt -installApp을 사용하여 USB를 통해 응용 프로그램을 설치할 수 있습니다.

6 idb -devices 명령을 사용하여 장치 핸들을 파악합니다(IDB는 ***air_sdk_root***/lib/aot/bin/iOSBin/idb에 있음).

```
./idb -devices
                                    List of attached devices
                                    Handle    UUID
                                        1     91770d8381d12644df91fbcee1c5bbdacb735500
```

참고: (AIR 3.4 이상)idb -devices 대신 adt -devices를 사용하여 장치 핸들을 파악할 수 있습니다.

7 IDB 유틸리티와 이전 단계에서 파악한 장치 ID를 사용하여 데스크톱의 포트를 adt -listen 매개 변수(이 경우 16000, 기본값은 7936)에서 지정된 포트로 전달합니다.

```
idb -forward 7936 16000 1
```

이 예제에서 7936은 데스크톱 포트이고, 16000은 연결된 장치가 수신 대기하는 포트이며, 1은 연결된 장치의 장치 ID입니다.

8 터미널 또는 명령 윈도우에서 -p 옵션을 사용하여 FDB를 실행합니다.

```
fdb -p 7936
```

9 FDB 윈도우에서 run 명령을 입력합니다.

```
Adobe fdb (Flash Player Debugger) [build 23201]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
```

10 FDB 유틸리티가 응용 프로그램에 연결하려고 시도합니다.

11 원격 연결이 설정되면 FDB break 명령을 사용하여 중단점을 설정한 후에 continue 명령을 사용하여 실행을 시작할 수 있습니다.

참고: 포트 번호 7936은 AIR 런타임과 FDB 모두에서 USB 디버깅의 기본값으로 사용됩니다. IDB -listen 포트 매개 변수 및 FDB -p 포트 매개 변수를 통해 사용할 다른 포트를 지정할 수 있습니다.

휴대 장치에 AIR 및 AIR 응용 프로그램 설치

응용 프로그램의 최종 사용자는 장치의 일반 응용 프로그램 및 배포 메커니즘을 사용하여 AIR 런타임 및 AIR 응용 프로그램을 설치할 수 있습니다.

예를 들어 Android에서는 사용자가 Android Market으로부터 응용 프로그램을 설치할 수 있습니다. 또는 응용 프로그램 설정에서 알 수 없는 소스로부터 응용 프로그램을 설치할 수 있도록 허용한 경우 웹 페이지의 링크를 클릭하거나, 응용 프로그램 패키지를 장치로 복사하여 열어봄으로써 응용 프로그램을 설치할 수 있습니다. 사용자가 Android 응용 프로그램을 설치하려고 하는데 아직 AIR 런타임이 설치되어 있지 않은 경우에는 런타임을 설치할 수 있는 Market으로 자동 이동합니다.

iOS에서는 두 가지 방법으로 응용 프로그램을 최종 사용자에게 배포할 수 있습니다. 기본 배포 채널은 Apple App Store입니다. 애드혹 배포를 사용하여 제한된 수의 사용자가 App Store를 거치지 않고도 귀하의 응용 프로그램을 설치하도록 허용할 수 있습니다.

개발용 AIR 런타임 및 응용 프로그램 설치

휴대 장치의 AIR 응용 프로그램은 기본 패키지로 설치되기 때문에 일반 플랫폼 기능을 사용하여 테스트용 응용 프로그램을 설치할 수 있습니다. 지원되는 경우, ADT 명령을 사용하여 AIR 런타임 및 AIR 응용 프로그램을 설치할 수 있습니다. 현재는 Android에서 이 방법이 지원됩니다.

iOS에서는 iTunes를 사용하여 테스트용 응용 프로그램을 설치할 수 있습니다. 테스트 응용 프로그램은 응용 프로그램 개발을 위해 특별히 발행되었고 개발 프로비저닝 프로파일과 함께 패키지화된 Apple 코드 서명 인증서를 사용하여 서명해야 합니다. AIR 응용 프로그램은 iOS에서 자체 포함 패키지이며 별도의 런타임이 사용되지 않습니다.

ADT를 사용하여 AIR 응용 프로그램 설치

AIR 응용 프로그램을 개발하는 동안 ADT를 사용하여 런타임과 응용 프로그램을 모두 설치 및 제거할 수 있습니다. ADT를 직접 실행해야 할 필요가 없도록 IDE에 이러한 명령이 통합되어 있을 수도 있습니다.

AIR ADT 유틸리티를 사용하여 장치 또는 에뮬레이터에 AIR 런타임을 설치할 수 있습니다. 장치에 대해 제공된 SDK를 설치해야 합니다. -installRuntime 명령을 사용하십시오.

```
adt -installRuntime -platform android -device deviceID -package path-to-runtime
```

-package 매개 변수가 지정되어 있지 않으면 설치된 AIR SDK에서 사용 가능한 것 중에서 장치 또는 에뮬레이터에 적합한 런타임 패키지가 선택됩니다.

Android 또는 iOS(AIR 3.4 이상)에 AIR 응용 프로그램을 설치하려면 유사한 -installApp 명령을 사용합니다.

```
adt -installApp -platform android -device deviceID -package path-to-app
```

-platform 인수에 대해 설정된 값은 설치 중인 장치와 일치해야 합니다.

참고: 기존 버전의 AIR 런타임 또는 AIR 응용 프로그램을 제거한 후에 다시 설치해야 합니다.

iTunes를 사용하여 iOS 장치에 AIR 응용 프로그램 설치

테스트를 위해 iOS 장치에 AIR 응용 프로그램을 설치하려면

1. iTunes 응용 프로그램을 엽니다.
2. 이 응용 프로그램의 프로비저닝 프로파일을 iTunes에 추가합니다. iTunes에서 [파일] > [보관함에 추가]를 선택합니다. 그런 다음 프로비저닝 프로파일 파일(파일 유형이 mobileprovision인 파일)을 선택합니다.
3. 일부 iTunes 버전은 같은 버전의 응용 프로그램이 이미 설치되어 있으면 응용 프로그램을 바꾸지 않습니다. 이 경우 장치와 iTunes의 응용 프로그램 목록에서 응용 프로그램을 삭제하십시오.
4. 응용 프로그램의 IPA 파일을 두 번 클릭합니다. 해당 파일이 iTunes에서 응용 프로그램 목록에 나타납니다.
5. 장치를 컴퓨터의 USB 포트에 연결합니다.
6. iTunes에서 장치의 [응용 프로그램] 탭에 있는 설치할 응용 프로그램 목록에 응용 프로그램이 선택되어 있는지 확인합니다.
7. iTunes 응용 프로그램의 왼쪽 목록에서 장치를 선택합니다. 그런 다음 [동기화] 버튼을 클릭합니다. 동기화가 완료되면 iPhone에 Hello World 응용 프로그램이 나타납니다.

새 버전이 설치되지 않은 경우 장치와 iTunes의 응용 프로그램 목록에서 응용 프로그램을 삭제하고 이 절차를 다시 실행합니다. 현재 설치된 버전의 응용 프로그램 ID와 버전이 동일하기 때문일 수 있습니다.

기타 도움말 항목

159페이지의 “ADT installRuntime 명령”

157페이지의 “ADT installApp 명령”

장치에서 AIR 응용 프로그램 실행

설치된 AIR 응용 프로그램을 장치 사용자 인터페이스를 사용하여 실행할 수 있습니다. 지원되는 경우, AIR ADT 유틸리티를 사용하여 응용 프로그램을 원격으로 실행할 수도 있습니다.

```
adt -launchApp -platform android -device deviceID -appid applicationID
```

-appid 인수의 값은 실행할 AIR 응용 프로그램의 AIR 응용 프로그램 ID여야 합니다. AIR 응용 프로그램 설명자에 지정된 값을 사용하십시오(패키지화 중에 추가된 **air** 접두어는 제외).

장치 또는 에뮬레이터가 하나만 연결되어 있고 실행 중이면 -device 플래그를 생략할 수 있습니다. -platform 인수에 대해 설정된 값은 설치 중인 장치와 일치해야 합니다. 현재는 **android**가 유일하게 지원되는 값입니다.

AIR 런타임 및 응용 프로그램 제거

장치 운영 체제에서 제공하는 일반적인 응용 프로그램 제거 방법을 사용할 수 있습니다. 지원되는 경우, AIR ADT 명령을 사용하여 AIR 런타임 및 응용 프로그램을 제거할 수도 있습니다. 런타임을 제거하려면 -uninstallRuntime 명령을 사용하십시오.

```
adt -uninstallRuntime -platform android -device deviceID
```

응용 프로그램을 제거하려면 -uninstallApp 명령을 사용하십시오.

```
adt -uninstallApp -platform android -device deviceID -appid applicationID
```

장치 또는 에뮬레이터가 하나만 연결되어 있고 실행 중이면 -device 플래그를 생략할 수 있습니다. -platform 인수에 대해 설정된 값은 설치 중인 장치와 일치해야 합니다. 현재는 **android**가 유일하게 지원되는 값입니다.

에뮬레이터 설정

장치 에뮬레이터에서 AIR 응용 프로그램을 실행하려면 일반적으로 장치에 대한 SDK를 사용하여 개발 컴퓨터에서 에뮬레이터 인스턴스를 만들고 실행해야 합니다. 그러면 AIR 런타임 및 AIR 응용 프로그램의 에뮬레이터 버전을 에뮬레이터에 설치할 수 있습니다. 에뮬레이터에 있는 응용 프로그램은 일반적으로 실제 장치에서보다 훨씬 느리게 실행됩니다.

Android 에뮬레이터 만들기

1 Android SDK 및 AVD Manager 응용 프로그램을 실행합니다.

- Windows에서는 Android SDK 디렉토리 루트에서 SDK Setup.exe 파일을 실행합니다.
- Mac OS에서는 Android SDK 디렉토리의 tools 하위 디렉토리에서 android 응용 프로그램을 실행합니다.

2 [Settings] 옵션을 선택하고 "Force https://" 옵션을 선택합니다.

3 [Available Packages] 옵션을 선택합니다. 사용 가능한 Android SDK의 목록이 표시됩니다.

4 호환되는 Android SDK(Android 2.3 이상)를 선택하고 [Install Selected] 버튼을 클릭합니다.

5 [Virtual Devices] 옵션을 선택하고 [New] 버튼을 클릭합니다.

6 다음 설정을 구성합니다.

- 가상 장치의 이름
- Android 2.3, API 레벨 8 등의 대상 API
- SD 카드의 크기(예: 1024)
- 스킨(예: 기본 HVGA)

7 [Create AVD] 버튼을 클릭합니다.

시스템 구성에 따라 가상 장치를 만드는 데 시간이 다소 걸릴 수 있습니다.

이제 새 가상 장치를 실행할 수 있습니다.

1 AVD Manager 응용 프로그램에서 [Virtual Device]를 선택합니다. 위에서 만든 가상 장치가 목록에 나타납니다.

2 [Virtual Device]를 선택하고 [Start] 버튼을 클릭합니다.

3 다음 화면에서 [Launch] 버튼을 클릭합니다.

데스크톱에서 에뮬레이터 윈도우가 열립니다. 시간이 몇 초 걸릴 수 있습니다. 또한 Android 운영 체제가 초기화되는 데 시간이 다소 걸릴 수도 있습니다. 에뮬레이터에서 **apk-debug** 및 **apk-emulator**를 사용하여 패키지화된 응용 프로그램을 설치할 수 있습니다. **apk** 대상으로 패키지화된 응용 프로그램은 에뮬레이터에서 작동하지 않습니다.

기타 도움말 항목

http://developer.android.com/guide/developing/tools/othertools.html#android

http://developer.android.com/guide/developing/tools/emulator.html

모바일 AIR 응용 프로그램 업데이트

모바일 AIR 응용 프로그램은 기본 패키지로 배포되므로 플랫폼에서 다른 응용 프로그램의 표준 업데이트 메커니즘을 사용하십시오. 일반적으로 이 과정에서 원래 응용 프로그램을 배포하는 데 사용한 것과 동일한 마켓플레이스 또는 App Store로 전송해야 합니다.

모바일 AIR 응용 프로그램은 AIR Updater 클래스 또는 프레임워크를 사용할 수 없습니다.

Android에서 AIR 응용 프로그램 업데이트

Android Market에 배포된 응용 프로그램의 경우에는 다음 사항을 모두 충족하면 Market에 새 버전을 전송하는 방식으로 응용 프로그램을 업데이트할 수 있습니다. 이 정책은 AIR가 아니라 Market을 통해 강제로 적용됩니다.

- APK 패키지가 동일한 인증서를 사용하여 서명되었습니다.
- AIR ID가 동일합니다.
- 응용 프로그램 설명자 파일에 있는 versionNumber 값이 더 큽니다. versionLabel 값이 사용되는 경우에는 이 값도 증가해야 합니다.

Android Market에서 귀하의 응용 프로그램을 다운로드한 사용자들은 업데이트를 받을 수 있다는 알림을 장치 소프트웨어로부터 받습니다.

기타 도움말 항목

Android 개발자: Android Market에서 업데이트 제작

iOS에서 AIR 응용 프로그램 업데이트

iTunes App Store를 통해 배포된 AIR 응용 프로그램의 경우, 다음 조건을 모두 충족하면 업데이트를 App Store로 전송하여 응용 프로그램을 업데이트할 수 있습니다(이러한 정책은 AIR가 아니라 Apple App Store에 의해 적용됨).

- 코드 서명 인증서 및 프로비저닝 프로파일이 같은 Apple ID에 대해 발급되었습니다.
- IPA 패키지가 같은 Apple Bundle ID를 사용합니다.
- 업데이트로 인해 지원되는 장치 풀이 줄어들지는 않습니다. 다시 말해, 원래 응용 프로그램이 iOS 3을 실행하는 장치를 지원하는 경우 iOS 3에 대한 지원이 삭제된 업데이트를 만들 수 없습니다.

중요: AIR SDK 버전 2.6 이상은 iOS 3을 지원하지 않고 AIR 2는 지원하기 때문에 AIR 2를 통해 개발된 기존 iOS 응용 프로그램을 AIR 2.6 이상을 통해 개발된 업데이트를 사용하여 업데이트할 수 없습니다.

푸시 알림 사용

푸시 알림은 원격 알림 공급자가 모바일 장치에서 실행되고 있는 응용 프로그램에 알림을 전송하도록 합니다. AIR 3.4에서는 APNs(Apple Push Notification service)를 사용하여 iOS 장치에 대해 푸시 알림을 지원합니다.

참고: AIR에서 Android 응용 프로그램에 대해 푸시 알림을 사용하려면 Adobe 전문가 Piotr Walczyszyn씨가 개발한 as3c2dm과 같은 기본 확장을 사용하십시오.

이 섹션의 나머지 부분에서는 AIR에서 iOS 응용 프로그램에 대해 푸시 알림을 활성화하는 방법을 설명합니다.

참고: 이 단원에서는 사용자가 Apple 개발자 ID를 가지고 있고 iOS 개발 작업 과정에 익숙하며, iOS 장치에 하나 이상의 응용 프로그램을 배포한 경험이 있다고 가정합니다.

푸시 알림 개요

APNs(Apple Push Notification service)는 원격 알림 공급자가 iOS 장치에서 실행되고 있는 응용 프로그램에 알림을 전송하도록 합니다. APNs에서 지원하는 알림 유형은 다음과 같습니다.

- 경고
- 배지
- 사운드

APNs에 대한 자세한 내용은 developer.apple.com을 참조하십시오.

응용 프로그램에서 푸시 알림을 사용하는 작업은 다음 여러 요소로 이루어집니다.

- **클라이언트 응용 프로그램** - 푸시 알림에 등록하고, 원격 알림 공급자와 통신하며, 푸시 알림을 수신합니다.
- **iOS** - 클라이언트 응용 프로그램과 APNs 간의 상호 작용을 관리합니다.
- **APNs** - 클라이언트 등록 과정에서 tokenID를 제공하고 원격 알림 공급자에서 iOS로 알림을 전달합니다.
- **원격 알림 공급자** - tokenId-클라이언트 응용 프로그램 정보를 저장하고 APNs로 알림을 푸시합니다.

등록 작업 과정

서버 측 서비스에 푸시 알림을 등록하는 작업 과정은 다음과 같습니다.

1. 클라이언트 응용 프로그램은 iOS에서 푸시 알림을 활성화하도록 요청합니다.
2. iOS에서 해당 요청을 APNs로 전달합니다.
3. APNs 서버에서 tokenId를 iOS에 반환합니다.
4. iOS에서 tokenId를 클라이언트 응용 프로그램에 반환합니다.
5. 클라이언트 응용 프로그램에서 응용 프로그램별 메커니즘을 통해 원격 알림 공급자에 tokenId를 제공하며, 원격 알림 공급자는 푸시 알림용으로 해당 tokenId를 저장합니다.

알림 작업 과정

알림 작업 과정은 다음과 같습니다.

1. 원격 알림 공급자에서 알림을 생성하고 해당 알림 페이로드를 tokenId와 함께 APNs에 전달합니다.
2. APNs에서 알림을 장치에 있는 iOS에 전달합니다.
3. iOS에서 알림 페이로드를 응용 프로그램에 푸시합니다.

푸시 알림 API

AIR 3.4에는 iOS 푸시 알림을 지원하는 API 세트가 추가되었습니다. 이 API는 flash.notifications 패키지에 들어 있으며 다음과 같은 클래스를 포함하고 있습니다.

- NotificationStyle - 알림 유형 ALERT, BADGE 및 SOUND.C에 대한 상수를 정의합니다.
- RemoteNotifier - 푸시 알림에 구독하거나 구독 해제하도록 합니다.
- RemoteNotifierSubscribeOptions - 수신할 알림 유형을 선택하도록 합니다. notificationStyles 속성을 사용하면 여러 알림 유형에 대해 등록할 문자열 벡터를 정의할 수 있습니다.

AIR 3.4에는 다음과 같이 RemoteNotifier에 의해 전달되는 flash.events.RemoteNotificationEvent도 포함되어 있습니다.

- 응용 프로그램의 구독이 성공적으로 작성되면 APNs로부터 새로운 tokenId를 수신하게 됩니다.
- 새 원격 알림을 수신하는 즉시 전달됩니다.

또한 RemoteNotifier는 구독 프로세스에서 오류가 발생할 경우 flash.events.StatusEvent를 전달합니다.

응용 프로그램 내 푸시 알림 관리

응용 프로그램에서 푸시 알림을 등록하려면 다음 단계를 수행하십시오.

- 응용 프로그램에서 푸시 알림에 구독하는 코드를 작성합니다.
- 응용 프로그램 XML 파일에서 푸시 알림을 활성화합니다.
- iOS 푸시 서비스를 활성화하는 프로비저닝 프로파일과 인증서를 만듭니다.

주석이 포함된 다음 샘플 코드는 푸시 알림에 구독하고 푸시 알림 이벤트를 처리하는 내용을 다룹니다.

```
package
                        {
                        import flash.display.Sprite;
                        import flash.display.StageAlign;
                        import flash.display.StageScaleMode;
                        import flash.events.*;
                        import flash.events.Event;
                        import flash.events.IOErrorEvent;
                        import flash.events.MouseEvent;
                        import flash.net.*;
                        import flash.text.TextField;
                        import flash.text.TextFormat;
                        import flash.ui.Multitouch;
                        import flash.ui.MultitouchInputMode;
                        // Required packages for push notifications
                        import flash.notifications.NotificationStyle;
                        import flash.notifications.RemoteNotifier;
                        import flash.notifications.RemoteNotifierSubscribeOptions;
                        import flash.events.RemoteNotificationEvent;
                        import flash.events.StatusEvent;
                        [SWF(width="1280", height="752", frameRate="60")]
                        public class TestPushNotifications extends Sprite
                        {
                        private var notiStyles:Vector.<String> = new Vector.<String>;;
                        private var tt:TextField = new TextField();
                        private var tf:TextFormat = new TextFormat();
                        // Contains the notification styles that your app wants to receive
                        private var preferredStyles:Vector.<String> = new Vector.<String>();
                        private var subscribeOptions:RemoteNotifierSubscribeOptions = new
RemoteNotifierSubscribeOptions();
                        private var remoteNot:RemoteNotifier = new RemoteNotifier();
                        private var subsButton:CustomButton = new CustomButton("Subscribe");
                        private var unSubsButton:CustomButton = new CustomButton("UnSubscribe");
                        private var clearButton:CustomButton = new CustomButton("clearText");
                        private var urlreq:URLRequest;
                        private var urlLoad:URLLoader = new URLLoader();
                        private var urlString:String;
                        public function TestPushNotifications()
                        {
                        super();
                        Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
                        stage.align = StageAlign.TOP_LEFT;
                        stage.scaleMode = StageScaleMode.NO_SCALE;
                        tf.size = 20;
                        tf.bold = true;
                        tt.x=0;
                        tt.y =150;
                        tt.height = stage.stageHeight;
                        tt.width = stage.stageWidth;
                        tt.border = true;
                        tt.defaultTextFormat = tf;
                        addChild(tt);
                        subsButton.x = 150;
                        subsButton.y=10;
                        subsButton.addEventListener(MouseEvent.CLICK,subsButtonHandler);
                        stage.addChild(subsButton);
                        unSubsButton.x = 300;
                        unSubsButton.y=10;
                        unSubsButton.addEventListener(MouseEvent.CLICK,unSubsButtonHandler);
                        stage.addChild(unSubsButton);
                        clearButton.x = 450;
                        clearButton.y=10;
```

```
                        clearButton.addEventListener(MouseEvent.CLICK,clearButtonHandler);
                        stage.addChild(clearButton);
                        //
                        tt.text += "\n SupportedNotification Styles: " +
RemoteNotifier.supportedNotificationStyles.toString() + "\n";
                        tt.text += "\n Before Preferred notificationStyles: " +
subscribeOptions.notificationStyles.toString() + "\n";
                        // Subscribe to all three styles of push notifications:
                        // ALERT, BADGE, and SOUND.
                        preferredStyles.push(NotificationStyle.ALERT
,NotificationStyle.BADGE,NotificationStyle.SOUND );
                        subscribeOptions.notificationStyles= preferredStyles;
                        tt.text += "\n After Preferred notificationStyles:" +
subscribeOptions.notificationStyles.toString() + "\n";
                        remoteNot.addEventListener(RemoteNotificationEvent.TOKEN,tokenHandler);

remoteNot.addEventListener(RemoteNotificationEvent.NOTIFICATION,notificationHandler);
                        remoteNot.addEventListener(StatusEvent.STATUS,statusHandler);
                        this.stage.addEventListener(Event.ACTIVATE,activateHandler);
                        }
                        // Apple recommends that each time an app activates, it subscribe for
                        // push notifications.
                        public function activateHandler(e:Event):void{
                        // Before subscribing to push notifications, ensure the device supports it.
                        // supportedNotificationStyles returns the types of notifications
                        // that the OS platform supports
                        if(RemoteNotifier.supportedNotificationStyles.toString() != "")
                        {
                        remoteNot.subscribe(subscribeOptions);
                        }
                        else{
                        tt.appendText("\n Remote Notifications not supported on this Platform !");
                        }
                        }
                        public function subsButtonHandler(e:MouseEvent):void{
                        remoteNot.subscribe(subscribeOptions);
                        }
                        // Optionally unsubscribe from push notfications at runtime.
                        public function unSubsButtonHandler(e:MouseEvent):void{
                        remoteNot.unsubscribe();
                        tt.text +="\n UNSUBSCRIBED";
                        }
                        public function clearButtonHandler(e:MouseEvent):void{
                        tt.text = " ";
                        }
                        // Receive notification payload data and use it in your app
                        public function notificationHandler(e:RemoteNotificationEvent):void{
                        tt.appendText("\nRemoteNotificationEvent type: " + e.type +
                        "\nbubbles: "+ e.bubbles + "\ncancelable " +e.cancelable);
                        for (var x:String in e.data) {
                        tt.text += "\n"+ x + ":  " + e.data[x];
                        }
                        }
                        // If the subscribe() request succeeds, a RemoteNotificationEvent of
                        // type TOKEN is received, from which you retrieve e.tokenId,
                        // which you use to register with the server provider (urbanairship, in
                        // this example.
                        public function tokenHandler(e:RemoteNotificationEvent):void
                        {
                        tt.appendText("\nRemoteNotificationEvent type: "+e.type +"\nBubbles: "+ e.bubbles
+ "\ncancelable " +e.cancelable +"\ntokenID:\n"+ e.tokenId +"\n");
                        urlString = new String("https://go.urbanairship.com/api/device_tokens/" +
```

```
e.tokenId);
urlreq = new URLRequest(urlString);
urlreq.authenticate = true;
urlreq.method = URLRequestMethod.PUT;
URLRequestDefaults.setLoginCredentialsForHost
("go.urbanairship.com",
"1ssB2iV_RL6_UBLiYMQVfg","t-kZlzXGQ6-yU8T3iHiSyQ");
urlLoad.load(urlreq);
urlLoad.addEventListener(IOErrorEvent.IO_ERROR,iohandler);
urlLoad.addEventListener(Event.COMPLETE,compHandler);
urlLoad.addEventListener(HTTPStatusEvent.HTTP_STATUS,httpHandler);
}
private function iohandler(e:IOErrorEvent):void
{
tt.appendText("\n In IOError handler" + e.errorID +" " +e.type);
}
private function compHandler(e:Event):void{
tt.appendText("\n In Complete handler,"+"status: " +e.type + "\n");
}
private function httpHandler(e:HTTPStatusEvent):void{
tt.appendText("\n in httpstatus handler,"+ "Status: " + e.status);
}
// If the subscription request fails, StatusEvent is dispatched with
// error level and code.
public function statusHandler(e:StatusEvent):void{
tt.appendText("\n statusHandler");
tt.appendText("event Level" + e.level +"\nevent code " +
e.code + "\ne.currentTarget: " + e.currentTarget.toString());
}
}
}
```

응용 프로그램 XML 파일에서 푸시 알림 활성화

응용 프로그램에서 푸시 알림을 사용하려면 iphone 태그 아래의 Entitlements 태그 내용을 다음과 같이 입력합니다.

```
<iphone>
                              ...
                                 <Entitlements>
                                    <![CDATA[
                                       <key>aps-environment</key>
                                       <string>development</string>
                                    ]]>
                                 </Entitlements>
                              </iphone>
```

응용 프로그램을 App Store에 추가할 준비가 완료되면 <string> 요소의 값을 development에서 production으로 변경합니다.

```
<string>production</string>
```

응용 프로그램에서 지역화된 문자열을 지원하는 경우 intialWindow 태그 아래의 supportedLanguages 태그에서 다음 예제와 같이 해당 언어를 지정하십시오.

```
<supportedLanguages>en de cs es fr it ja ko nl pl pt</supportedLanguages>
```

iOS 푸시 서비스를 활성화하는 프로비저닝 프로파일과 인증서 만들기

응용 프로그램 및 APNs 간 통신을 활성화하려면 해당 응용 프로그램을 iOS 푸시 서비스를 활성화하는 프로비저닝 프로파일 및 인증서와 함께 패키지화해야 합니다.

1 Apple 개발자 계정으로 로그인합니다.

2 Provisioning Portal로 이동합니다.

3 [응용 프로그램 ID] 탭을 클릭합니다.

4 [새 응용 프로그램 ID] 버튼을 클릭합니다.

5 설명 및 번들 식별자를 지정합니다. 번들 식별자에는 * 기호를 사용하지 않습니다.

6 [전송]을 클릭합니다. Provisioning Portal에서 응용 프로그램 ID를 생성하고 응용 프로그램 ID 페이지를 다시 표시합니다.

7 응용 프로그램 ID 오른쪽에 있는 [구성]을 클릭합니다. [응용 프로그램 ID 구성] 페이지가 표시됩니다.

8 [Apple 푸시 알림 서비스 사용] 체크 상자를 선택합니다. 푸시 SSL 인증서에는 각각 개발/테스트용과 실제 업무용으로 두 가지 유형이 있습니다.

9 개발용 푸시 SSL 인증서의 오른쪽에 있는 [구성] 버튼을 클릭합니다. [인증서 서명 요청(CSR) 생성] 페이지가 표시됩니다.

10 페이지의 지침에 따라 키체인 접근 유틸리티를 사용하여 CSR을 생성합니다.

11 SSL 인증서를 생성합니다.

12 SSL 인증서를 다운로드하여 설치합니다.

13 (선택 사항) 실제 업무용 푸시 SSL 인증서에 대해 9~12단계를 반복합니다.

14 [완료]를 클릭합니다. [응용 프로그램 ID 구성] 페이지가 표시됩니다.

15 [완료]를 클릭합니다. [응용 프로그램 ID] 페이지가 표시됩니다. 해당 응용 프로그램 ID에 대한 푸시 알림 옆에 있는 녹색 원을 주목하십시오.

16 SSL 인증서를 저장합니다. 이 인증서는 나중에 응용 프로그램 및 공급자 간 통신에 사용됩니다.

17 [프로비저닝] 탭을 클릭하여 [프로비저닝 프로파일] 페이지를 표시합니다.

18 새 응용 프로그램 ID에 대한 프로비저닝 프로파일을 작성한 후 다운로드합니다.

19 [인증서] 탭을 클릭하여 새 프로비저닝 프로파일에 대한 새 인증서를 다운로드합니다.

푸시 알림에 사운드 사용

응용 프로그램에서 사운드 알림을 사용하려면 SWF 및 app-xml 파일과 같은 디렉토리에 다른 에셋의 경우처럼 사운드 파일을 번들합니다. 예를 들면 다음과 같습니다.

```
Build/adt -package -target ipa-app-store -provisioning-profile _-_.mobileprovision -storetype pkcs12 -
keystore _-_.p12 test.ipa test-app.xml test.swf sound.caf sound1.caf
```

Apple에서 지원하는 사운드 데이터 형식(aiff, wav 또는 caf 파일 형태)은 다음과 같습니다.

- Linear PCM
- MA4(IMA/ADPCM)
- uLaw
- aLaw

지역화된 경고 알림 사용

응용 프로그램에서 지역화된 경고 알림을 사용하려면 지역화된 문자열을 lproj 폴더의 형식으로 번들하십시오. 예를 들어 경고를 스페인어로 지원하려면 다음과 같이 합니다.

1 프로젝트 안에 app-xml 파일과 같은 수준에서 es.lproj 폴더를 만듭니다.

2 es.lproj 폴더 안에서 Localizable.Strings라는 이름으로 텍스트 파일을 만듭니다.

3 텍스트 편집기에서 Localizable.Strings를 열고 메시지 키와 해당되는 지역화된 문자열을 추가합니다. 예를 들면 다음과 같습니다.

```
"PokeMessageFormat" = "La notificación de alertas en español."
```

4 파일을 저장합니다.

5 응용 프로그램에서 이 키 값을 갖는 경고 알림을 수신하고 장치 언어가 스페인어인 경우 변환된 경고 텍스트가 표시됩니다.

원격 알림 공급자 구성

응용 프로그램에 푸시 알림을 보내려면 원격 알림 공급자가 필요합니다. 이 서버 응용 프로그램은 사용자의 푸시 입력을 허용하고 알림 및 알림 데이터를 APNs에 전달하는 공급자로 작동하며, 곧이어 APNs에서는 푸시 알림을 클라이언트 응용 프로그램에 전송합니다.

원격 알림 공급자에서 알림을 푸시하는 방식에 대한 자세한 내용은 Apple 개발자 라이브러리에서 공급자와 APNs(Apple Push Notification service) 간 통신을 참조하십시오.

원격 알림 공급자 옵션

원격 알림 공급자에 대한 옵션은 다음과 같습니다.

- APNS-php 오픈 소스 서버를 기반으로 사용자 고유의 공급자를 만듭니다. http://code.google.com/p/apns-php/의 내용을 참고하여 PHP 서버를 설정할 수 있습니다. 이 Google 코드 프로젝트를 활용하면 사용자의 특정 요구 사항을 충족하는 인터페이스를 설계할 수 있습니다.
- 서비스 공급자를 사용합니다. 예를 들어 http://urbanairship.com/에서는 미리 준비된 APNs 공급자를 제공합니다. 이 서비스에 등록한 후에는 가장 먼저 다음과 비슷한 코드를 사용하여 장치 토큰을 제공합니다.

```
private var urlreq:URLRequest;
                                    private var urlLoad:URLLoader = new URLLoader();
                                    private var urlString:String;
                                    //When subscription is successful then only call the following code
                                    urlString = new
String("https://go.urbanairship.com/api/device_tokens/" + e.tokenId);
                                    urlreq = new URLRequest(urlString);
                                    urlreq.authenticate = true;
                                    urlreq.method = URLRequestMethod.PUT;

URLRequestDefaults.setLoginCredentialsForHost("go.urbanairship.com",
                                       "Application Key","Application Secret");
                                    urlLoad.load(urlreq);
                                    urlLoad.addEventListener(IOErrorEvent.IO_ERROR,iohandler);
                                    urlLoad.addEventListener(Event.COMPLETE,compHandler);

urlLoad.addEventListener(HTTPStatusEvent.HTTP_STATUS,httpHandler);
                                    private function iohandler(e:IOErrorEvent):void{
                                       trace("\n In IOError handler" + e.errorID +" " +e.type);
                                    }
                                    private function compHandler(e:Event):void{
                                       trace("\n In Complete handler,"+"status: " +e.type + "\n");
                                    }
                                    private function httpHandler(e:HTTPStatusEvent):void{
                                       tt.appendText("\n in httpstatus handler,"+ "Status: " +
e.status);
                                    }
```

그런 다음 Urban Airship 도구를 사용하여 테스트 알림을 전송할 수 있습니다.

원격 알림 공급자에 대한 인증서

원격 알림 공급자의 서버에 있는 적절한 위치에 SSL 인증서와 이전에 생성된 개인 키를 복사해 놓아야 합니다. 이 두 파일은 하나의 .pem 파일로 통합하는 것이 일반적입니다. 이렇게 하려면 다음 단계를 수행하십시오.

1 터미널 윈도우를 엽니다.

2 다음 명령을 입력하여 SSL 인증서로부터 .pem 파일을 생성합니다.

```
openssl x509 -in aps_developer_identity.cer -inform der -out TestPushDev.pem
```

3 다음 명령을 입력하여 개인 키 파일의 .pem 파일을 생성(.p12)합니다.

```
openssl pkcs12 -nocerts -out TestPushPrivateKey.pem -in certificates.p12
```

4 다음 명령을 입력하여 두 .pem 파일을 단일 파일로 결합합니다.

```
cat TestPushDev.pem TestPushPrivateKey.pem > FinalTestPush.pem
```

5 서버 측 푸시 응용 프로그램을 만들 때, 결합된 .pem 파일을 서버 공급자에 제공합니다.

자세한 내용은 Apple 로컬 및 푸시 알림 프로그래밍 안내서에서 서버에 SSL 인증서 및 키 설치를 참조하십시오.

응용 프로그램 내 푸시 알림 처리

응용 프로그램 내의 푸시 알림 처리는 다음 작업으로 이루어집니다.

- 전역 사용자 구성 및 푸시 알림의 허용
- 개별 푸시 알림에 대한 사용자의 허용
- 푸시 알림 및 알림 페이로드 데이터의 처리

구성 및 푸시 알림의 허용

iOS에서는 사용자가 푸시 알림 지원 응용 프로그램을 처음으로 시작할 때 [허용 안 함] 및 [확인] 버튼이 포함된 **appname 푸시 알림 전송** 대화 상자를 표시합니다. 사용자가 [확인]을 선택하면 응용 프로그램에서 구독하는 모든 스타일의 알림을 수신할 수 있습니다. [허용 안 함]을 선택하면 아무런 알림도 수신되지 않습니다.

참고: [설정] > [알림]으로 이동하면 각 푸시 지원 응용 프로그램과 관련해 수신할 수 있는 특정 알림 유형을 제어할 수 있습니다.

Apple에서는 응용 프로그램이 활성화될 때마다 푸시 알림에 구독하도록 설정할 것을 권장하고 있습니다. 응용 프로그램에서 RemoteNotifier.subscribe()를 호출하면 token 유형의 RemoteNotificationEvent를 수신하게 되며, 여기에는 장치에서 해당 응용 프로그램을 고유하게 식별하는 32바이트 고유 숫자 tokenId가 포함되어 있습니다.

장치에서 푸시 알림을 수신하면 [닫기] 및 [시작] 버튼이 포함된 팝업이 표시됩니다. 사용자가 [닫기]를 터치하면 아무런 동작도 발생하지 않습니다. [시작]을 터치하면 iOS에서는 해당 응용 프로그램을 호출하고 이 응용 프로그램은 아래에 설명된 것처럼 notification 유형의 flash.events.RemoteNotificationEvent를 수신합니다.

푸시 알림 및 페이로드 데이터의 처리

원격 알림 공급자에서 tokenID를 사용하여 장치에 알림을 전송하면 응용 프로그램에서는 현재 실행 여부에 관계없이 notification 유형의 flash.events.RemoteNotificationEvent를 수신하게 됩니다. 응용 프로그램은 이 시점에서 응용 프로그램별 알림 처리를 수행합니다. 응용 프로그램에서 알림 데이터를 처리할 경우 사용자는 해당 데이터를 JSON 형식의 RemoteNotificationEvent.data 속성을 통해 액세스할 수 있습니다.

8장: TV 장치용 AIR 응용 프로그램 개발

TV용 AIR 기능

해당 장치에 TV용 Adobe AIR이 포함되어 있을 경우 TV, 디지털 비디오 레코더, Blu-ray 플레이어 등 TV 장치용 Adobe® AIR® 응용 프로그램을 만들 수 있습니다. TV용 AIR은 장치의 하드웨어 액셀러레이터를 고성능 비디오 및 그래픽에 사용하는 등 TV 장치에 최적화되어 있습니다.

TV 장치용 AIR 응용 프로그램은 HTML 기반이 아닌 SWF 기반 응용 프로그램입니다. TV용 AIR 응용 프로그램은 하드웨어 가속뿐만 아니라 "거실" 환경에 적합한 다른 AIR 기능도 활용할 수 있습니다.

장치 프로파일

AIR는 프로파일을 사용하여 비슷한 기능의 대상 장치 집합을 정의합니다. TV용 AIR 응용 프로그램에 대해 다음 프로파일을 사용합니다.

- tv 프로파일. TV용 AIR 장치를 대상으로 하는 AIR 응용 프로그램에서 이 프로파일을 사용합니다.
- extendedTV 프로파일. TV용 AIR 응용 프로그램에서 기본 확장을 사용하는 경우 이 프로파일을 사용합니다.

이들 프로파일에 대해 정의된 ActionScript 기능은 222페이지의 “장치 프로파일”에 설명되어 있습니다. TV용 AIR 응용 프로그램의 구체적인 ActionScript 차이점은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에 설명되어 있습니다.

TV용 AIR 프로파일에 대한 자세한 내용은 127페이지의 “지원되는 프로파일”을 참조하십시오.

하드웨어 가속

TV 장치는 AIR 응용 프로그램에서 그래픽 및 비디오의 성능을 대폭 향상시키는 하드웨어 액셀러레이터를 제공합니다. 이러한 하드웨어 액셀러레이터의 이점을 활용하려면 111페이지의 “TV용 AIR 응용 프로그램 설계 고려 사항”을 참조하십시오.

내용 보호

TV용 AIR에서는 할리우드 블록버스터부터 독립 영화 및 TV 프로그램에 이르기까지 프리미엄 비디오 내용에 대한 풍부한 소비자 환경을 구축할 수 있습니다. 내용 공급업체는 Adobe의 도구를 사용하여 대화형 응용 프로그램을 만들 수 있습니다. Adobe 서버 제품을 자사의 내용 배포 인프라에 통합하거나, Adobe의 에코시스템 파트너와 협력할 수도 있습니다.

내용 보호는 프리미엄 비디오 배포의 핵심 요구 사항입니다. TV용 AIR는 주요 영화 제작업체를 비롯한 내용 소유자들의 엄격한 보안 요구 사항을 충족하는 내용 보호 및 결제 솔루션인 Adobe® Flash® Access™를 지원합니다.

Flash Access는 다음을 지원합니다.

- 비디오 스트리밍 및 다운로드
- 광고 지원, 가입, 임대, 전자 판매 등의 다양한 비즈니스 모델
- HTTP Dynamic Streaming, Flash® Media Server를 사용한 RTMP(Real Time Media Protocol)상의 스트리밍, HTTP를 통한 점진적 다운로드 같은 여러 내용 전송 기술

TV용 AIR는 보안 요구 사항이 보다 낮은 기존 스트리밍 솔루션을 위한 RTMPE(암호화된 버전의 RTMP)도 기본적으로 지원합니다. RTMPE 및 관련 SWF 확인 기술은 Flash Media Server에서 지원됩니다.

자세한 내용은 Adobe Flash Access를 참조하십시오.

다중 채널 오디오

AIR 3부터 TV용 AIR는 HTTP 서버에서 점진적으로 다운로드되는 비디오에 대해 다중 채널 오디오를 지원합니다. 이러한 지원에는 다음과 같은 코덱이 포함됩니다.

- AC-3(Dolby Digital)
- E-AC-3(Enhanced Dolby Digital)
- DTS Digital Surround
- DTS Express
- DTS-HD High Resolution Audio
- DTS-HD Master Audio

참고: Adobe Flash Media Server에서 스트리밍되는 비디오의 다중 채널 오디오 지원은 아직 제공되지 않습니다.

게임 입력

AIR 3부터 TV용 AIR는 응용 프로그램에서 조이스틱, 게임 패드, 막대 같은 연결된 게임 입력 장치와 통신할 수 있도록 하는 ActionScript API를 지원합니다. 이러한 장치를 게임 입력 장치라고는 하지만 단순히 게임뿐 아니라 모든 TV용 AIR 응용 프로그램에서 장치를 사용할 수 있습니다.

다양한 기능을 갖춘 광범위한 게임 입력 장치를 사용할 수 있습니다. 따라서 응용 프로그램이 각기 다른(또는 알려지지 않은) 유형의 게임 입력 장치와 올바르게 작동할 수 있도록 장치가 API에서 일반화됩니다.

GameInput 클래스는 게임 입력 ActionScript API에 대한 진입점 역할을 하는 클래스입니다. 자세한 내용은 GameInput을 참조하십시오.

스테이지 3D 가속 그래픽 렌더링

AIR 3부터 TV용 AIR는 스테이지 3D 가속 그래픽 렌더링을 지원합니다. Stage3D ActionScript API는 고급 2D 및 3D 기능을 구현하는 하위 수준 GPU 가속 API 집합입니다. 개발자는 이러한 하위 수준 API를 통해 보다 자유롭게 GPU 하드웨어 가속을 활용함으로써 성능을 대폭 끌어올릴 수 있습니다. 또한 Stage3D ActionScript API를 지원하는 게임 엔진을 사용할 수도 있습니다.

자세한 내용은 게임 엔진, 3D 및 스테이지 3D를 참조하십시오.

기본 확장

응용 프로그램이 extendedTV 프로파일을 대상으로 하는 경우 ANE(AIR Native Extension) 패키지를 사용할 수 있습니다.

일반적으로 장치 제조업체는 AIR에서 지원하지 않는 장치 기능에 액세스할 수 있도록 하기 위해 ANE 패키지를 제공합니다. 예를 들어 기본 확장을 통해 TV의 채널을 변경하거나 비디오 플레이어의 재생을 일시 중지할 수 있습니다.

ANE 패키지를 사용하는 TV용 AIR 응용 프로그램을 패키지화할 때는 응용 프로그램을 AIR 파일 대신 AIRN 파일로 패키지화합니다.

TV용 AIR 장치에 대한 기본 확장은 항상 **장치에 번들로 제공되는** 기본 확장입니다. 장치에 번들로 제공된다는 것은 TV용 AIR 장치에 확장 라이브러리가 설치되어 있다는 의미입니다. 응용 프로그램 패키지에 포함하는 ANE 패키지에는 확장의 기본 라이브러리가 포함되어 있지 **않습니다**. 경우에 따라서는 ActionScript 전용 버전의 기본 확장이 포함되어 있습니다. 이 ActionScript 전용 버전은 확장의 스텁 또는 시뮬레이터입니다. 장치 제조업체에서는 기본 라이브러리 등의 실제 확장을 장치에 설치합니다.

기본 확장을 개발하는 경우 다음 사항에 유의합니다.

- 장치에 대한 TV용 AIR 기본 확장을 만드는 경우에는 항상 장치 제조업체에 문의합니다.
- 일부 TV용 AIR 장치의 경우에는 장치 제조업체에서만 기본 확장을 만듭니다.

- 모든 TV용 AIR 장치의 경우 장치 제조업체에서는 설치할 기본 확장을 결정합니다.
- TV용 AIR 기본 확장을 만드는 데 사용되는 개발 도구는 제조업체마다 다릅니다.

AIR 응용 프로그램에서 기본 확장을 사용하는 방법에 대한 자세한 내용은 134페이지의 “Adobe AIR용 기본 확장 사용”을 참조하십시오.

기본 확장을 만드는 방법에 대한 자세한 내용은 Adobe AIR용 기본 확장 개발을 참조하십시오.

TV용 AIR 응용 프로그램 설계 고려 사항

비디오 고려 사항

비디오 인코딩 지침

비디오를 TV 장치로 스트리밍할 때 Adobe는 다음과 같은 인코딩 지침을 권장합니다.

비디오 코덱:	H.264, Main 또는 High 프로파일, 점진적 인코딩
해상도:	720i, 720p, 1080i 또는 1080p
프레임 속도:	초당 24프레임 또는 초당 30프레임
오디오 코덱:	AAC-LC 또는 AC-3, 44.1kHz, 스테레오 또는 이러한 다중 채널 오디오 코덱: E-AC-3, DTS, DTS Express, DTS-HD High Resolution Audio 또는 DTS-HD Master Audio
결합 비트율:	사용 가능한 대역폭에 따라 최대 8Mbps
오디오 비트율:	최대 192Kbps
픽셀 종횡비:	1 × 1

TV용 AIR 장치에 제공되는 비디오에 대해서는 H.264 코덱을 사용하는 것이 좋습니다.

참고: TV용 AIR는 Sorenson Spark 또는 On2 VP6 코덱을 사용하여 인코딩된 비디오도 지원합니다. 하지만 하드웨어는 이러한 코드를 디코딩하여 표현하지 않습니다. 대신 런타임은 소프트웨어를 사용하여 이러한 코드를 디코딩하여 표현하므로 비디오는 훨씬 낮은 프레임 속도로 재생됩니다. 따라서 가능하면 언제나 H.264를 사용하십시오.

StageVideo 클래스

TV용 AIR에서는 하드웨어가 H.264로 인코딩된 비디오를 디코딩하여 표현합니다. 이 기능을 활성화하려면 StageVideo 클래스를 사용하십시오.

다음에 대한 자세한 내용은 **ActionScript 3.0 개발자 가이드**에서 하드웨어 가속 표현에 StageVideo 클래스 사용을 참조하십시오.

- StageVideo 클래스 및 관련 클래스의 API
- StageVideo 클래스의 사용 제한

H.264 인코딩된 비디오에 대해 Video 객체를 사용하는 기존 AIR 응용 프로그램을 최선으로 지원하기 위해 TV용 AIR는 내부에서 StageVideo 객체를 사용합니다. 따라서 비디오 재생 시 하드웨어로 디코딩 및 표현하는 이점을 누리게 됩니다. 하지만 Video 객체는 StageVideo 객체와 동일한 제한을 받습니다. 예를 들어 응용 프로그램이 비디오를 회전하려고 할 때 런타임이 아닌 하드웨어가 비디오를 표현하기 때문에 회전이 이루어지지 않습니다.

하지만 새 응용 프로그램을 작성할 때는 H.264 인코딩된 비디오에 대해 StageVideo 객체를 사용하십시오.

StageVideo 클래스의 사용 예는 TV에서 Flash 플랫폼을 위한 비디오 및 내용 제공을 참조하십시오.

비디오 전송 지침

TV용 AIR 장치에서는 비디오 재생 중에 네트워크의 사용 가능한 대역폭에 변화가 있을 수 있습니다. 예를 들어 다른 사용자가 같은 인터넷 연결을 사용하기 시작할 때 이러한 변화가 발생할 수 있습니다.

따라서 비디오 전송 시스템에서 적응형 비트율 기능을 사용하는 것이 좋습니다. 예를 들어 서버측에서는 Flash Media Server가 적응형 비트율 기능을 지원합니다. 클라이언트측에서는 OSMF(Open Source Media Framework)를 사용할 수 있습니다.

다음 프로토콜을 사용하여 비디오 내용을 네트워크를 통해 TV용 AIR 응용 프로그램으로 전송할 수 있습니다.

- HTTP 및 HTTPS Dynamic Streaming(F4F 형식)
- RTMP, RTMPE, RTMFP, RTMPT 및 RTMPTE 스트리밍
- HTTP 및 HTTPS 점진적 다운로드

자세한 내용은 다음을 참조하십시오.

- Adobe Flash Media Server 개발자 안내서
- Open Source Media Framework

오디오 고려 사항

TV용 AIR 응용 프로그램에서 사운드 재생을 위한 ActionScript는 다른 AIR 응용 프로그램에서와 마찬가지입니다. 자세한 내용은 **ActionScript 3.0** 개발자 가이드에서 사운드 작업을 참조하십시오.

TV용 AIR에서의 다중 채널 오디오 지원에 대해서는 다음 사항을 고려합니다.

- TV용 AIR는 HTTP 서버에서 점진적으로 다운로드되는 비디오에 대해 다중 채널 오디오를 지원합니다. Adobe Flash Media Server에서 스트리밍되는 비디오의 다중 채널 오디오 지원은 아직 제공되지 않습니다.
- TV용 AIR에서는 많은 오디오 코덱을 지원하지만 모든 TV용 AIR 장치에서 전체 세트를 지원하는 것은 아닙니다. flash.system.Capabilities 메서드 hasMultiChannelAudio()를 사용하여 TV용 AIR 장치가 AC-3 같은 특정 다중 채널 오디오 코덱을 지원하는지 여부를 확인합니다.

  예를 들어 서버에서 비디오 파일을 점진적으로 다운로드하는 응용 프로그램을 가정해 봅니다. 서버에는 서로 다른 다중 채널 오디오 코덱을 지원하는 여러 H.264 비디오 파일이 있습니다. 응용 프로그램은 hasMultiChannelAudio()를 사용하여 서버에서 요청할 비디오 파일을 결정할 수 있습니다. 또는 응용 프로그램은 Capabilities.serverString에 포함된 문자열을 서버에 보낼 수 있습니다. 문자열은 서버에서 적절한 비디오 파일을 선택할 수 있도록 사용 가능한 다중 채널 오디오 코덱을 나타냅니다.
- DTS 오디오 코덱 중 하나를 사용할 때는 hasMultiChannelAudio()가 true를 반환하지만 DTS 오디오가 재생되지 않는 경우가 있습니다.

  예를 들어 S/PDIF 출력이 있는 Blu-ray 플레이어를 구식 앰프에 연결한 경우를 가정해 봅니다. 구식 앰프는 DTS를 지원하지 않지만 Blu-ray 플레이어에 이를 알려줄 프로토콜이 S/PDIF에 없습니다. Blu-ray 플레이어가 구식 앰프에 DTS 스트림을 보내면 사용자에게 아무 것도 들리지 않습니다. 따라서 DTS를 사용할 때는 사운드가 재생되지 않는 경우를 사용자가 알릴 수 있도록 사용자 인터페이스를 제공하는 것이 좋습니다. 이렇게 하면 응용 프로그램에서 다른 코덱으로 전환할 수 있습니다.

다음 표에는 TV용 AIR 응용 프로그램에서 서로 다른 오디오 코덱을 사용해야 할 경우가 요약되어 있습니다. 또한 이 표에는 TV용 AIR 장치가 하드웨어 액셀러레이터를 사용하여 오디오 코덱을 디코딩하는 경우가 나와 있습니다. 하드웨어 디코딩은 성능을 향상시키고 CPU의 부하를 줄여 줍니다.

오디오 코덱	TV용 AIR 장치에서의 가용성	하드웨어 디코딩	이 오디오 코덱을 사용해야 할 때	추가 정보
AAC	항상	항상	H.264로 인코딩된 비디오에서 사용 인터넷 음악 스트리밍 서비스 등의 오디오 스트리밍에 사용	오디오 전용 AAC 스트림을 사용할 때는 오디오 스트림을 MP4 컨테이너에 캡슐화하십시오.
mp3	항상	아니요	응용 프로그램의 SWF 파일에 있는 사운드에 사용 Sorenson Spark 또는 On2 VP6로 인코딩된 비디오에서 사용	오디오에 mp3를 사용하는 H.264 비디오는 TV용 AIR 장치에서 재생되지 않습니다.
AC-3(Dolby Digital) E-AC-3(Enhanced Dolby Digital) DTS Digital Surround DTS Express DTS-HD High Resolution Audio DTS-HD Master Audio	확인	예	H.264로 인코딩된 비디오에서 사용	일반적으로 TV용 AIR는 오디오를 디코딩 및 재생하는 외부 오디오/비디오 수신기에 다중 채널 오디오 스트림을 전달합니다.
Speex	항상	아니요	라이브 음성 스트림을 받을 때 사용	오디오에 Speex를 사용하는 H.264 비디오는 TV용 AIR 장치에서 재생되지 않습니다. Speex는 Sorenson Spark 또는 On2 VP6로 인코딩된 비디오에만 사용합니다.
NellyMoser	항상	아니요	라이브 음성 스트림을 받을 때 사용	오디오에 NellyMoser를 사용하는 H.264 비디오는 TV용 AIR 장치에서 재생되지 않습니다. NellyMoser는 Sorenson Spark 또는 On2 VP6로 인코딩된 비디오에만 사용합니다.

참고: 일부 비디오 파일에는 두 개의 오디오 스트림이 포함되어 있습니다. 예를 들어 비디오 파일은 AAC 스트림 및 AC3 스트림을 모두 포함할 수 있습니다. TV용 AIR는 이러한 비디오 파일을 지원하지 않으며 이러한 파일을 사용하면 비디오의 사운드가 재생되지 않을 수 있습니다.

그래픽 하드웨어 가속

하드웨어 그래픽 가속 사용

TV용 AIR 장치는 2D 그래픽 작업에 대해 하드웨어 가속을 제공합니다. 장치의 하드웨어 그래픽 액셀러레이터는 다음 작업을 수행해야 하는 CPU의 부담을 덜어 줍니다.

- 비트맵 렌더링
- 비트맵 크기 조절
- 비트맵 블렌딩
- 단색 사각형 채우기

이 하드웨어 그래픽 가속은 TV용 AIR 응용 프로그램에서 여러 그래픽 작업을 높은 성능으로 수행할 수 있음을 의미합니다. 이러한 기능 중 일부는 다음과 같습니다.

- 슬라이딩 전환

- 크기 조절 전환
- 페이드 인 및 페이드 아웃
- 알파를 사용하여 복수 이미지 혼합

이러한 유형의 작업에 대해 하드웨어 그래픽 가속의 성능 이점을 누리려면 다음 기법 중 하나를 사용하십시오.

- MovieClip 객체 그리고 대부분 변경되지 않는 내용을 포함하고 있는 다른 표시 객체에 대해서는 cacheAsBitmap 속성을 true로 설정합니다. 그런 다음 해당 객체에 대해 슬라이딩 전환, 페이딩 전환, 알파 블렌딩을 수행합니다.
- 크기를 조절하거나 변환(x 및 y 재배치 적용)할 표시 객체에 대해 cacheAsBitmapMatrix 속성을 사용합니다.

  크기 조절 및 변환에 Matrix 클래스 작업을 사용하면 장치의 하드웨어 액셀러레이터가 작업을 수행합니다. 또는 cacheAsBitmap 속성이 true로 설정된 표시 객체의 크기를 변경하는 시나리오를 생각해 보십시오. 크기가 변경되면 런타임의 소프트웨어가 비트맵을 다시 그립니다. 소프트웨어를 사용하여 다시 그리는 것은 Matrix 작업을 사용하여 하드웨어 가속으로 크기를 조절하는 것보다 성능이 떨어집니다.

  예를 들어 최종 사용자가 선택할 때 확장되는 이미지를 표시하는 응용 프로그램을 생각해 보십시오. Matrix 크기 조절 작업을 여러 번 사용하여 이미지가 확장되는 느낌을 줍니다. 하지만 원래 이미지 및 최종 이미지의 크기에 따라 최종 이미지의 품질이 많이 떨어질 수 있습니다. 따라서 확장 작업이 완료된 후에 표시 객체의 크기를 재설정하십시오. cacheAsBitmap이 true이기 때문에 런타임 소프트웨어는 표시 객체를 다시 그리지만 한 번만 다시 그리고 고품질 이미지로 렌더링합니다.

  참고: 일반적으로 TV용 AIR 장치는 하드웨어 가속화된 회전 및 기울이기를 지원하지 않습니다. 따라서 Matrix 클래스에서 회전 및 기울이기를 지정하는 경우 TV용 AIR는 모든 Matrix 작업을 소프트웨어에서 수행합니다. 이러한 소프트웨어 작업은 성능에 좋지 않은 영향을 줄 수 있습니다.
- BitmapData 클래스를 통해 사용자 정의 비트맵 캐싱 비헤이비어를 만듭니다.

비트맵 캐싱에 대한 자세한 내용은 다음을 참조하십시오.

- 표시 객체 캐싱
- 비트맵 캐싱
- 수동 비트맵 캐싱

그래픽 메모리 관리

가속화된 그래픽 작업을 수행하기 위해 하드웨어 액셀러레이터가 특수 그래픽 메모리를 사용합니다. 응용 프로그램에서 모든 그래픽 메모리를 사용하는 경우에는 TV용 AIR가 소프트웨어를 사용하여 그래픽 작업을 수행하는 방식으로 되돌아가기 때문에 응용 프로그램이 더 느리게 실행됩니다.

응용 프로그램의 그래픽 메모리 사용을 관리하려면

- 이미지 또는 다른 비트맵 데이터의 사용을 마쳤으면 관련 그래픽 메모리를 릴리스합니다. 이렇게 하려면 Bitmap 객체의 bitmapData 속성의 dispose() 메서드를 호출합니다. 예를 들면 다음과 같습니다.

  ```
  myBitmap.bitmapData.dispose();
  ```

  참고: BitmapData 객체에 대한 참조를 릴리스해도 그래픽 메모리가 바로 비워지지는 않습니다. 결국 런타임의 가비지 컬렉터가 그래픽 메모리를 비우기는 하지만 dispose()를 호출하면 응용 프로그램의 제어권이 강화됩니다.
- Adobe에서 제공하는 AIR 응용 프로그램인 PerfMaster Deluxe를 사용하면 대상 장치에서 하드웨어 그래픽 가속에 대해 보다 정확하게 이해할 수 있습니다. 이 응용 프로그램은 다양한 작업을 실행하기 위한 초당 프레임 수를 보여 줍니다. PerfMaster Deluxe를 사용하여 서로 다르게 구현된 같은 작업을 비교해 보십시오. 예를 들어 비트맵 이미지를 이동하는 경우와 벡터 이미지를 이동하는 경우를 비교해 보십시오. PerfMaster Deluxe는 TV용 Flash 플랫폼에서 받을 수 있습니다.

표시 목록 관리

표시 객체를 숨기려면 객체의 visible 속성을 false로 설정합니다. 그러면 객체는 여전히 표시 목록에 있지만 TV용 AIR에서는 이를 렌더링하거나 표시하지 않습니다. 이 기법은 처리 오버헤드가 적게 발생하기 때문에 뷰를 자주 왕복하는 객체에 유용합니다. 그러나 visible 속성을 false로 설정하더라도 객체의 리소스가 해제되지는 않습니다. 따라서 객체의 표시를 완료했거나, 최소한 객체를 오랫동안 사용하여 작업을 완료한 경우에는 표시 목록에서 해당 객체를 제거합니다. 또한 객체에 대한 모든 참조를 null로 설정합니다. 이러한 작업을 수행하면 가비지 수집기를 통해 객체의 리소스가 해제됩니다.

PNG 및 JPEG 이미지 사용

응용 프로그램에서 사용되는 두 개의 공통된 이미지 형식은 PNG와 JPEG입니다. TV용 AIR 응용 프로그램에서는 이러한 두 이미지 형식과 관련하여 다음 사항을 고려하십시오.

- TV용 AIR는 일반적으로 하드웨어 가속을 사용하여 JPEG 파일을 디코딩합니다.
- TV용 AIR는 대개 소프트웨어를 사용하여 PNG 파일을 디코딩합니다. 소프트웨어에서 PNG 파일을 디코딩하는 것이 빠릅니다.
- PNG는 유일하게 플랫폼에 영향을 받지 않는 비트맵 형식으로서 투명도(알파 채널)를 지원합니다.

따라서 응용 프로그램에서 이러한 이미지 형식을 다음과 같이 사용하십시오.

- 하드웨어 가속화된 디코딩을 활용할 수 있도록 사진에는 JPEG 파일을 사용합니다.
- 사용자 인터페이스 요소에는 PNG 이미지 파일을 사용합니다. 사용자 인터페이스 요소에는 알파 설정이 있을 수 있으며, 소프트웨어 디코딩은 사용자 인터페이스 요소에 충분한 성능을 제공합니다.

TV용 AIR 응용 프로그램의 스테이지

TV용 AIR 응용 프로그램을 만드는 중에 Stage 클래스로 작업할 때는 다음 사항을 고려하십시오.

- 화면 해상도
- 안전 보기 영역
- 스테이지 크기 조절 모드
- 스테이지 정렬
- 스테이지 표시 상태
- 여러 스크린 크기에 맞춰 디자인
- 스테이지 품질 설정

화면 해상도

현재 TV 장치는 일반적으로 540p, 720p, 1080p 화면 해상도 중 하나를 사용합니다. 이러한 화면 해상도는 ActionScript Capabilities 클래스에서 다음과 같은 값을 갖게 됩니다.

화면 해상도	Capabilities.screenResolutionX	Capabilities.screenResolutionY
540p	960	540
720p	1280	720
1080p	1920	1080

특정 장치에 대한 전체 화면 TV용 AIR 응용 프로그램을 작성하려면 Stage.stageWidth 및 Stage.stageHeight를 장치의 화면 해상도에 맞춰 하드 코딩하십시오. 하지만 여러 장치에서 실행되는 전체 화면 응용 프로그램을 작성하려면 Capabilities.screenResolutionX 및 Capabilities.screenResolutionY 속성을 사용하여 Stage 크기를 설정하십시오.

예를 들면 다음과 같습니다.

```
stage.stageWidth = Capabilities.screenResolutionX;
stage.stageHeight = Capabilities.screenResolutionY;
```

안전 보기 영역

TV에서 **안전 보기 영역**이란 스크린 가장자리로부터 약간의 공간을 남겨 놓는 화면 영역입니다. 이 영역은 TV의 베젤이 화면의 어느 부분도 가리지 않아서 최종 사용자가 전체 영역을 볼 수 있을 정도로 충분한 간격으로 두고 안쪽으로 들어와 있습니다. 화면 주위의 물리적 프레임인 베젤은 제조업체마다 다르기 때문에 필요한 간격도 각기 다릅니다. 안전 보기 영역은 화면의 가시 영역을 보장하기 위한 것입니다. 안전 보기 영역을 **제목 보호 영역**이라고도 합니다.

오버스캔은 베젤 뒤에 있어서 보이지 않는 화면 영역입니다.

각 화면 가장자리에는 7.5%의 간격을 두는 것이 좋습니다. 예를 들면 다음과 같습니다.

화면 해상도 1920 x 1080의 안전 보기 영역

전체 화면 TV용 AIR 응용 프로그램을 디자인할 때는 항상 안전 보기 영역을 고려하십시오.

- 배경 이미지나 배경색 등의 배경에는 전체 화면을 사용합니다.
- 텍스트, 그래픽, 비디오, 사용자 인터페이스 항목(예: 버튼) 등의 중요한 응용 프로그램 요소에만 안전 보기 영역을 사용합니다.

다음 표에는 7.5%의 간격을 사용할 때 일반적인 각 화면 해상도의 안전 보기 영역 크기가 나와 있습니다.

화면 해상도	안전 보기 영역의 폭과 높이	왼쪽 및 오른쪽 간격 폭	위쪽 및 아래쪽 간격 폭
960 x 540	816 x 460	72	40
1280 x 720	1088 x 612	96	54
1920 x 1080	1632 x 918	144	81

하지만 가장 좋은 방법은 항상 안전 보기 영역을 동적으로 계산하는 것입니다. 예를 들면 다음과 같습니다.

```
var horizontalInset, verticalInset, safeAreaWidth, safeAreaHeight:int;

horizontalInset = .075 * Capabilities.screenResolutionX;
verticalInset = .075 * Capabilities.screenResolutionY;
safeAreaWidth = Capabilities.screenResolutionX - (2 * horizontalInset);
safeAreaHeight = Capabilities.screenResolutionY - (2 * verticalInset);
```

스테이지 크기 조절 모드

Stage.scaleMode를 StageScaleMode.NO_SCALE로 설정하고 Stage resize 이벤트를 기다립니다.

```
stage.scaleMode = StageScaleMode.NO_SCALE;
stage.addEventListener(Event.RESIZE, layoutHandler);
```

이 설정은 스테이지 좌표를 픽셀 좌표와 동일하게 만듭니다. FULL_SCREEN_INTERACTIVE 표시 상태 및 TOP_LEFT 스테이지 정렬과 함께 이 설정을 적용하면 안전 보기 영역을 효과적으로 사용할 수 있습니다.

특히 전체 화면 응용 프로그램에서 이 크기 조절 모드는 Stage 클래스의 stageWidth 및 stageHeight 속성이 Capabilities 클래스의 screenResolutionX 및 screenResolutionY 속성에 해당함을 의미합니다.

아울러 응용 프로그램 윈도우 크기가 변경될 때 스테이지 내용은 정해진 크기를 유지합니다. 런타임에서는 자동 레이아웃 또는 크기 조절을 수행하지 않습니다. 또한 런타임은 윈도우 크기가 변경될 때 Stage 클래스의 resize 이벤트를 전달합니다. 따라서 응용 프로그램이 시작될 때 그리고 응용 프로그램 윈도우 크기가 변경될 때 응용 프로그램의 내용을 원하는 대로 조정할 수 있습니다.

참고: NO_SCALE 비헤이비어는 다른 AIR 응용 프로그램에서와 마찬가지입니다. 하지만 TV용 AIR 응용 프로그램에서는 안전 보기 영역을 사용할 때 이 설정을 사용하는 것이 중요합니다.

스테이지 정렬

Stage.align을 StageAlign.TOP_LEFT로 설정합니다.

```
stage.align = StageAlign.TOP_LEFT;
```

이 정렬은 ActionScript를 사용하여 내용을 배치할 때 편리하도록 0,0 좌표를 화면의 왼쪽 위 모서리에 배치합니다.

NO_SCALE 크기 조절 모드 및 FULL_SCREEN_INTERACTIVE 표시 상태와 함께 이 설정을 적용하면 안전 보기 영역을 효과적으로 사용할 수 있습니다.

스테이지 표시 상태

전체 화면 TV용 AIR 응용 프로그램에서 Stage.displayState를 StageDisplayState.FULL_SCREEN_INTERACTIVE로 설정합니다.

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

이 값은 AIR 응용 프로그램이 사용자 입력을 허용한 상태에서 전체 화면에 걸쳐 스테이지를 확장하도록 설정합니다.

가급적 FULL_SCREEN_INTERACTIVE 설정을 사용하는 것이 좋습니다. NO_SCALE 크기 조절 모드 및 TOP_LEFT 스테이지 정렬과 함께 이 설정을 적용하면 안전 보기 영역을 효과적으로 사용할 수 있습니다.

따라서 전체 화면 응용 프로그램의 경우 주 문서 클래스에 대한 ADDED_TO_STAGE 이벤트의 핸들러에서 다음을 수행합니다.

```
private function onStage(evt:Event):void
{
    stage.scaleMode = StageScaleMode.NO_SCALE;
    stage.align = StageAlign.TOP_LEFT;
    stage.addEventListener(Event.RESIZE, onResize);
    stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
}
```

그런 다음 RESIZE 이벤트의 핸들러에서 다음을 수행합니다.

- 화면 해상도 크기를 스테이지 폭 및 높이와 비교합니다. 동일한 경우에는 스테이지 표시 상태가 FULL_SCREEN_INTERACTIVE로 변경되었기 때문에 RESIZE 이벤트가 발생한 것입니다.
- 안전 보기 영역 그리고 해당하는 간격의 크기를 계산하여 저장합니다.

```
private function onResize(evt:Event):void
{
    if ((Capabilities.screenResolutionX == stage.stageWidth) &&
        (Capabilities.screenResolutionY == stage.stageHeight))
    {

        // Calculate and save safe viewing area dimensions.
    }
}
```

스테이지 크기가 Capabilities.screenResolutionX 및 screenResolutionY와 같은 경우 TV용 AIR에서는 하드웨어에서 해당 비디오 및 그래픽에 대해 최상의 품질을 구현하도록 합니다.

참고: 그래픽 및 비디오가 TV 화면에 표시되는 품질은 Capabilities.screenResolutionX 및 screenResolutionY 값과 다를 수 있는데, 이러한 값은 TV용 AIR이 실행되는 장치에 따라 달라집니다. 예를 들어 TV용 AIR을 실행하는 셋탑 박스의 화면 해상도는 1280 x 720이고 연결된 TV의 화면 해상도는 1920 x 1080일 수 있습니다. 하지만 TV용 AIR에서는 하드웨어에서 최상의 품질을 구현하도록 지시하지 않습니다. 따라서 이 예의 경우 하드웨어는 1920 x 1080 화면 해상도를 사용하여 1080p 비디오를 표시합니다.

여러 스크린 크기에 맞춰 디자인

같은 전체 화면 TV용 AIR 응용 프로그램이 여러 TV용 AIR 장치에서 제대로 작동하고 올바르게 표시되도록 개발할 수 있습니다. 다음을 수행하십시오.

1. 스테이지 속성 scaleMode, align 및 displayState를 권장 값으로 설정합니다. 권장되는 값은 각각 StageScaleMode.NO_SCALE, StageAlign.TOP_LEFT, StageDisplayState.FULL_SCREEN_INTERACTIVE입니다.
2. Capabilities.screenResolutionX 및 Capabilities.screenResolutionY를 기준으로 안전 보기 영역을 설정합니다.
3. 안전 보기 영역의 폭과 높이에 맞춰 내용의 크기와 레이아웃을 조절합니다.

   특히 휴대 장치 응용 프로그램과 비교할 경우 내용의 객체가 크지만 동적 레이아웃, 상대적 배치, 적응형 내용 등의 개념은 동일합니다. 이러한 개념을 지원하는 ActionScript에 대한 자세한 내용은 여러 화면 크기에 대한 모바일 Flash 내용 만들기를 참조하십시오.

스테이지 품질

TV용 AIR 응용 프로그램에 대한 Stage.quality 속성은 항상 StageQuality.High이며, 이를 변경할 수는 없습니다.

이 속성은 모든 Stage 객체의 렌더링 품질을 지정합니다.

리모컨 입력 처리

사용자는 일반적으로 리모컨을 사용하여 TV용 AIR 응용 프로그램과 상호 작용합니다. 하지만 데스크톱 응용 프로그램에서 키보드의 키 입력을 처리할 때와 동일한 방식으로 키 입력을 처리하십시오. 특히 KeyboardEvent.KEY_DOWN 이벤트를 처리하십시오. 자세한 내용은 **ActionScript 3.0** 개발자 가이드에서 키보드 입력 캡처를 참조하십시오.

리모컨의 키는 ActionScript 상수에 매핑됩니다. 예를 들어 리모컨의 방향 키패드에 있는 키는 다음과 같이 매핑됩니다.

리모컨의 방향 키패드 키	ActionScript 3.0 상수
위로	Keyboard.UP
아래로	Keyboard.DOWN
왼쪽	Keyboard.LEFT
오른쪽	Keyboard.RIGHT
OK 또는 Select	Keyboard.ENTER

AIR 2.5에서는 리모컨 입력을 지원하기 위한 여러 Keyboard 상수가 추가되었습니다. 전체 목록은 **Adobe Flash Platform용 ActionScript 3.0 참조 설명서**에서 Keyboard 클래스를 참조하십시오.

최대한 많은 장치에서 응용 프로그램이 작동하도록 하려면 다음 권장 사항을 따르는 것이 좋습니다.

- 가능한 경우 방향 키패드 키만 사용하십시오.

  리모컨 장치마다 키 집합이 서로 다릅니다. 하지만 일반적으로 방향 키패드 키는 항상 있습니다.

  예를 들어 Blu-ray 플레이어의 리모컨에는 일반적으로 "채널 위로" 및 "채널 아래로" 키가 없습니다. 재생, 일시 중지, 중지 등의 키가 없는 리모컨도 있습니다.

- 응용 프로그램에서 방향 키패드 키보다 더 많은 키가 필요한 경우에는 Menu 및 Info 키를 사용하십시오.

  Menu 및 Info 키는 리모컨에서 그 다음으로 가장 흔한 키입니다.

- 범용 리모컨이 널리 사용된다는 점을 고려하십시오.

  특정 장치를 위한 응용 프로그램을 만들더라도 많은 사용자들은 장치와 함께 제공되는 리모컨을 사용하는 대신 범용 리모컨을 사용합니다. 또한 장치의 리모컨에 있는 모든 키와 일치하도록 범용 리모컨을 프로그래밍하지 않는 경우가 많습니다. 따라서 가장 흔한 키만 사용하는 것이 좋습니다.

- 사용자가 방향 키패드 키 중 하나를 사용하여 모든 상황을 종료할 수 있도록 하십시오.

  때로는 응용 프로그램이 리모컨에서 가장 흔한 키가 아닌 다른 키를 사용해야 할 합당한 이유가 있을 수 있습니다. 방향 키패드 중 하나를 사용하여 모든 것을 취소할 수 있도록 만들면 응용 프로그램이 모든 장치에서 원활하게 작동합니다.

- 대상 TV용 AIR 장치에 포인터 입력 기능이 있음을 알고 있는 경우를 제외하면 포인터 입력을 요구하지 마십시오.

  많은 데스크톱 응용 프로그램에서는 마우스 입력을 사용해야 하지만 대부분의 TV에서는 포인터 입력을 지원하지 않습니다. 따라서 TV에서 실행되도록 데스크톱 응용 프로그램을 변환 중인 경우에는 마우스 입력이 필요 없도록 응용 프로그램을 수정하십시오 이벤트 처리 및 사용자 지침을 변경하는 것도 잊지 마십시오. 예를 들어 응용 프로그램의 시작 화면이 표시될 때 "시작하려면 클릭하십시오."라는 텍스트가 표시되지 않도록 하십시오.

포커스 관리

데스크톱 응용 프로그램에서 한 사용자 인터페이스 요소에 포커스가 있으면 키보드 및 마우스 이벤트 등의 사용자 입력 이벤트가 해당 요소를 대상으로 실행됩니다. 또한 응용 프로그램은 포커스가 있는 사용자 인터페이스를 강조 표시합니다. 다음과 같은 이유로 인해 TV용 AIR 응용 프로그램에서 포커스를 관리하는 것은 데스크톱 응용 프로그램에서 포커스를 관리하는 것과 다릅니다.

- 데스크톱 응용 프로그램은 종종 Tab 키를 사용하여 다음 사용자 인터페이스 요소로 포커스를 변경합니다. TV용 AIR 응용 프로그램에는 Tab 키 사용이 적용되지 않습니다. 리모컨 장치에는 일반적으로 Tab 키가 없습니다. 따라서 데스크톱에서 DisplayObject의 tabEnabled 속성을 사용하여 포커스를 관리하는 방식이 적용되지 않습니다.
- 데스크톱 응용 프로그램에서는 종종 사용자가 마우스를 사용하여 사용자 인터페이스 요소에 포커스를 지정해야 합니다.

따라서 응용 프로그램에서 다음을 수행하십시오.

- KeyboardEvent.KEY_DOWN 등의 Keyboard 이벤트를 수신 대기하는 이벤트 리스너를 Stage에 추가합니다.
- 최종 사용자에게 강조 표시할 사용자 인터페이스 요소를 결정할 응용 프로그램 논리를 제공합니다. 응용 프로그램이 시작될 때 사용자 인터페이스 요소를 강조 표시합니다.
- 응용 프로그램 논리를 바탕으로 Stage가 해당하는 사용자 인터페이스 요소 객체로부터 받은 Keyboard 이벤트를 전달합니다.

  Stage.focus 또는 Stage.assignFocus()를 사용하여 사용자 인터페이스 요소에 포커스를 할당할 수도 있습니다. 그런 다음 해당 DisplayObject에 이벤트 리스너를 추가하여 키보드 이벤트를 받도록 할 수 있습니다.

사용자 인터페이스 디자인

다음과 관련된 권장 사항을 통합하여 TV용 AIR 응용 프로그램의 사용자 인터페이스가 TV에서 제대로 작동하도록 만드십시오.

- 응용 프로그램의 응답성
- 응용 프로그램의 사용 편의성
- 사용자의 개성 및 기대치

응답성

다음 팁에 따라 TV용 AIR 응용 프로그램이 최대한 빠르게 응답하도록 만드십시오.

- 응용 프로그램의 초기 SWF 파일을 최대한 작게 만듭니다.

  초기 SWF 파일에서 응용 프로그램을 시작하는 데 필요한 리소스만 로드하십시오. 예를 들어 응용 프로그램의 시작 화면 이미지만 로드합니다.

  이 권장 사항은 데스크톱 AIR 응용 프로그램에도 유효하지만 TV용 AIR 장치에서 더욱 중요합니다. 여기에는 여러 가지 이유가 있습니다. 우선 TV용 AIR 장치는 데스크톱 컴퓨터보다 처리 능력이 떨어집니다. 또한 응용 프로그램을 플래시 메모리에 저장합니다. 플래시 메모리는 데스크톱 컴퓨터의 하드 디스크만큼 빨리 액세스할 수 없습니다.

- 응용 프로그램이 최소한 초당 20프레임의 프레임 속도로 실행되도록 만듭니다.

  이 목표에 맞춰 그래픽을 디자인하십시오. 그래픽 작업이 복잡하면 초당 프레임 수에 영향을 줄 수 있습니다. 렌더링 성능을 높이는 방법에 대한 팁은 Adobe Flash 플랫폼의 성능 최적화를 참조하십시오.

  참고: TV용 AIR 장치의 그래픽 하드웨어는 일반적으로 60Hz 또는 120Hz(초당 60회 또는 120회)의 속도로 화면을 업데이트합니다. 하드웨어는 60Hz 또는 120Hz 화면에 표시하기 위해 초당 30프레임 또는 초당 60프레임 등의 속도로 스테이지에서 업데이트를 스캔합니다. 하지만 응용 프로그램의 그래픽 작업이 복잡하면 사용자가 이보다 더 높은 프레임 속도를 경험할 수 있습니다.

- 사용자 입력 후 100~200밀리초 이내에 업데이트하십시오.

  업데이트가 오래 걸리는 경우 사용자가 참지 못하고 키를 여러 번 누르는 경우가 종종 발생합니다.

사용 편의성

TV용 AIR 응용 프로그램의 사용자는 "거실" 환경에 있습니다. TV에서 3미터 정도 떨어진 거리에 앉아 있는 것입니다. 때로는 실내가 어두울 수도 있습니다. 그리고 주로 리모컨을 사용하여 입력합니다. 여러 명이 응용 프로그램을 사용할 수 있습니다. 때로는 함께 사용할 수도, 때로는 연달아 사용할 수도 있습니다.

따라서 TV에서 사용하기 편하도록 사용자 인터페이스를 디자인하려면 다음 사항을 고려하십시오.

- 사용자 인터페이스 요소를 크게 만듭니다.

  텍스트, 버튼 또는 기타 다른 사용자 인터페이스 요소를 디자인할 때 사용자가 실내의 반대편에 있다는 점을 고려하십시오. 3미터 정도의 거리에서 모든 것을 쉽게 보고 읽을 수 있도록 만드십시오. 화면이 크다고 해서 화면을 꽉 채워서는 안 됩니다.

- 실내의 반대편에서 내용을 쉽게 보고 읽을 수 있도록 대비를 활용합니다.
- 포커스가 있는 사용자 인터페이스 요소를 밝게 표시하여 명확하게 알 수 있도록 합니다.
- 필요할 때만 모션을 사용합니다. 예를 들어 연속성을 위해 한 화면에서 다음 화면으로 슬라이드처럼 넘어가는 것은 좋은 방법입니다. 하지만 사용자가 탐색하는 데 도움이 되지 않거나 응용 프로그램과 관련된 것이 아닐 경우에는 모션이 방해가 될 수 있습니다.
- 항상 사용자 인터페이스를 뒤로 이동할 수 있는 명확한 방법을 제공합니다.

리모컨의 사용에 대한 자세한 내용은 118페이지의 "리모컨 입력 처리"를 참조하십시오.

사용자의 개성 및 기대치

TV용 AIR 응용 프로그램의 사용자는 일반적으로 재미있고 편안한 환경에서 TV 품질의 엔터테인먼트를 원합니다. 컴퓨터 또는 기술에 대해 잘 모를 수도 있습니다.

따라서 다음과 같은 특징을 갖도록 TV용 AIR 응용 프로그램을 디자인하십시오.

- 기술적인 용어를 사용하지 않습니다.
- 가능하면 모달 대화 상자를 사용하지 않습니다.
- 업무 또는 기술 환경이 아니라 거실 환경에 알맞은 친숙하고 이해하기 쉬운 지침을 사용합니다.
- TV 시청자가 기대하는 높은 품질의 그래픽을 사용합니다.
- 리모컨으로 손쉽게 작동하는 사용자 인터페이스를 만듭니다. 데스크톱 또는 모바일 응용 프로그램에 더 적합한 사용자 인터페이스 또는 디자인 요소는 사용하지 마십시오. 예를 들어 데스크톱 및 모바일 장치의 사용자 인터페이스에서는 마우스나 손가락으로 버튼을 가리키고 클릭하는 동작을 수행해야 하는 경우가 많습니다.

글꼴 및 텍스트

TV용 AIR 응용 프로그램에서 장치 글꼴 또는 포함된 글꼴을 사용할 수 있습니다.

장치 글꼴은 장치에 설치된 글꼴입니다. 모든 TV용 AIR 장치에는 다음과 같은 장치 글꼴이 있습니다.

글꼴 이름	설명
_sans	_sans 장치 글꼴은 sans-serif 서체입니다. 모든 TV용 AIR 장치에 설치된 _sans 장치 글꼴은 Myriad Pro입니다. 일반적으로 sans-serif 서체는 보기 거리로 인해 serif 서체보다 TV에 더 잘 어울립니다.
_serif	_serif 장치 글꼴은 serif 서체입니다. 모든 TV용 AIR 장치에 설치된 _serif 장치 글꼴은 Minion Pro입니다.
_typewriter	_typewriter 장치 글꼴은 monospace 글꼴입니다. 모든 TV용 AIR 장치에 설치된 _typewriter 장치 글꼴은 Courier Std입니다.

모든 TV용 AIR 장치에는 다음과 같은 아시아 장치 글꼴도 있습니다.

글꼴 이름	언어	서체 범주	로캘 코드
RyoGothicPlusN-Regular	일본어	sans	ja
RyoTextPlusN-Regular	일본어	serif	ja
AdobeGothicStd-Light	한국어	sans	ko
AdobeHeitiStd-Regular	중국어 간체	sans	zh_CN
AdobeSongStd-Light	중국어 간체	serif	zh_CN
AdobeMingStd-Light	중국어 번체	serif	zh_TW 및 zh_HK

TV용 AIR 장치 글꼴의 특징은 다음과 같습니다.

- Adobe® Type Library에서 가져온 것입니다.
- TV에서 보기에 좋습니다.
- 비디오 타이틀에 알맞게 디자인되었습니다.
- 비트맵 글꼴이 아니라 글꼴 윤곽선입니다.

참고: 장치 제조업체는 장치에 다른 장치 글꼴도 포함하는 경우가 많습니다. 제조업체에서 제공한 이러한 장치 글꼴은 TV용 AIR 장치 글꼴에 더해 설치됩니다.

Adobe에서는 장치에 있는 모든 장치 글꼴을 표시하는 FontMaster Deluxe라는 응용 프로그램을 제공합니다. 이 응용 프로그램은 TV용 Flash 플랫폼에서 받을 수 있습니다.

TV용 AIR 응용 프로그램에 글꼴을 포함할 수도 있습니다. 포함된 글꼴에 대한 자세한 내용은 **ActionScript 3.0** 개발자 가이드에서 고급 텍스트 렌더링을 참조하십시오.

Adobe에서는 TLF 텍스트 필드와 관련하여 다음 사항을 권장합니다.

- 아시아 언어 텍스트에 대해 TLF 텍스트 필드를 사용하여 응용 프로그램이 실행되고 있는 로캘의 이점을 활용하십시오. TLFTextField 객체와 연관된 TextLayoutFormat 객체의 locale 속성을 설정하십시오. 현재 로캘을 파악하려면 **ActionScript 3.0** 개발자 가이드에서 로캘 선택을 참조하십시오.
- TV용 AIR 장치 글꼴이 아닌 글꼴을 사용하는 경우 TextLayoutFormat 객체의 fontFamily 속성에서 글꼴 이름을 지정하십시오. 장치에서 사용할 수 있는 경우 TV용 AIR에서는 해당 글꼴을 사용합니다. 요청한 글꼴이 locale 설정을 기준으로 장치에 없을 경우 TV용 AIR는 적절한 TV용 AIR 장치 글꼴로 대체합니다.
- fontFamily 속성에 대해 _sans, _serif 또는 _typewriter를 지정하고 locale 속성을 설정하여 TV용 AIR가 올바른 TV용 AIR 장치 글꼴을 선택하도록 합니다. 로캘에 따라 TV용 AIR는 아시아 장치 글꼴 집합 또는 비아시아 장치 글꼴 집합 중에서 선택합니다. 이 설정을 사용함으로써 4개의 주요 아시아 로캘 및 영어에 대해 올바른 글꼴을 자동으로 쉽게 사용할 수 있습니다.

참고: 아시아 언어 텍스트에 대해 클래식 텍스트 필드를 사용하는 경우 올바른 렌더링을 보장하기 위해 TV용 AIR 장치 글꼴의 글꼴 이름을 지정하십시오. 대상 장치에 다른 글꼴이 설치되어 있다는 것을 알고 있으면 해당 글꼴을 지정할 수도 있습니다.

응용 프로그램 성능과 관련하여 다음 사항을 고려하십시오.

- 클래식 텍스트 필드는 TLF 텍스트 필드보다 속도가 빠릅니다.
- 비트맵 글꼴을 사용하는 클래식 텍스트 필드가 가장 빠릅니다.

  비트맵 글꼴은 각 문자에 대한 윤곽선 데이터만 제공하는 윤곽선 글꼴과 달리 각 문자에 대한 비트맵을 제공합니다. 장치 글꼴과 포함된 글꼴 모두 비트맵 글꼴이 될 수 있습니다.
- 장치 글꼴을 지정하는 경우 해당 장치 글꼴이 대상 장치에 설치되어 있는지 확인하십시오. 장치에 설치되어 있지 않으면 TV용 AIR가 장치에 설치된 다른 글꼴을 찾아서 사용합니다. 하지만 이 비헤이비어는 응용 프로그램의 속도를 떨어뜨립니다.
- 표시 객체와 마찬가지로 TextField 객체가 대부분 변경되지 않는 경우에는 객체의 cacheAsBitmap 속성을 true로 설정하십시오. 이 설정은 페이드, 슬라이드, 알파 블렌딩 등 전환 성능을 향상시킵니다. 크기 조절 및 변환에는 cacheAsBitmapMatrix를 사용하십시오. 자세한 내용은 113페이지의 “그래픽 하드웨어 가속”을 참조하십시오.

파일 시스템 보안

TV용 AIR 응용 프로그램은 AIR 응용 프로그램이므로 장치의 파일 시스템에 액세스할 수 있습니다. 하지만 “거실” 장치에서는 응용 프로그램이 장치의 시스템 파일 또는 다른 응용 프로그램의 파일에 액세스할 수 없도록 만드는 것이 매우 중요합니다. TV 및 관련 장치의 사용자는 장치 오류를 예상하거나 용납하지 않습니다. 이들에게는 단지 TV를 보는 것일 뿐이기 때문입니다.

따라서 TV용 AIR 응용 프로그램에서는 장치의 파일 시스템을 제한적으로만 볼 수 있습니다. ActionScript 3.0을 사용하면 응용 프로그램이 특정 디렉토리(및 해당 하위 디렉토리)에만 액세스할 수 있습니다. 그뿐만 아니라 ActionScript에서 사용하는 디렉토리 이름은 장치에 있는 실제 디렉토리 이름이 아닙니다. 이러한 추가 레이어는 TV용 AIR 응용 프로그램이 해당 응용 프로그램에 속하지 않는 로컬 파일에 악의적으로 또는 실수로 액세스하는 것을 방지합니다.

자세한 내용은 TV용 AIR 응용 프로그램의 디렉토리 보기를 참조하십시오.

AIR 응용 프로그램 샌드박스

TV용 AIR 응용 프로그램은 AIR 응용 프로그램 샌드박스에서 실행됩니다. AIR 응용 프로그램 샌드박스에 대한 설명은 AIR 응용 프로그램 샌드박스를 참조하십시오.

TV용 AIR 응용 프로그램의 유일한 차이점은 122페이지의 "파일 시스템 보안"에 설명된 대로 파일 시스템에 대한 액세스가 제한되어 있다는 점입니다.

응용 프로그램 수명 주기

데스크톱 환경과 달리 최종 사용자는 TV용 AIR 응용 프로그램이 실행되고 있는 윈도우를 닫을 수 없습니다. 따라서 응용 프로그램을 종료하기 위한 사용자 인터페이스 메커니즘을 제공하십시오.

일반적으로 장치에서 최종 사용자는 리모컨의 종료 키를 사용하여 응용 프로그램을 무조건 종료할 수 있습니다. 하지만 TV용 AIR에서는 flash.events.Event.EXITING 이벤트를 응용 프로그램에 전달하지 않습니다. 따라서 다음에 시작될 때 응용 프로그램이 제대로 복원될 수 있도록 응용 프로그램 상태를 자주 저장하십시오.

HTTP 쿠키

TV용 AIR는 HTTP 영구 쿠키 및 세션 쿠키를 지원합니다. TV용 AIR는 각 AIR 응용 프로그램의 쿠키를 응용 프로그램별 디렉토리에 저장합니다.

`/app-storage/<app id>/Local Store`

쿠키 파일의 이름은 cookies입니다.

참고: 데스크톱 장치 같은 다른 장치의 AIR는 각 응용 프로그램에 대한 쿠키를 별도로 저장하지 않습니다. 응용 프로그램별 쿠키 저장소는 TV용 AIR의 응용 프로그램 및 시스템 보안 모델을 지원합니다.

ActionScript 속성 URLRequest.manageCookies를 다음과 같이 사용하십시오.

- manageCookies를 true로 설정합니다. 이 값은 기본값으로, TV용 AIR가 HTTP 요청에 쿠키를 자동으로 추가하고 HTTP 응답에서 쿠키를 기억한다는 것을 의미합니다.

  참고: manageCookies가 true일 때도 응용 프로그램은 URLRequest.requestHeaders를 사용하여 HTTP 요청에 쿠키를 수동으로 추가할 수 있습니다. 이 쿠키의 이름이 TV용 AIR에서 관리하는 쿠키의 이름과 똑같은 경우에는 같은 이름의 두 쿠키가 요청에 포함됩니다. 두 쿠키의 값은 서로 다를 수 있습니다.

- manageCookies를 false로 설정합니다. 이 값은 응용 프로그램이 HTTP 요청에서 쿠키를 보내고 HTTP 응답에서 쿠키를 기억하는 것을 담당한다는 것을 의미합니다.

자세한 내용은 URLRequest를 참조하십시오.

TV용 AIR 응용 프로그램 개발을 위한 작업 과정

다음과 같은 Adobe Flash Platform 개발 도구를 사용하여 TV용 AIR 응용 프로그램을 개발할 수 있습니다.

- Adobe Flash Professional

  Adobe Flash Professional CS5.5는 TV용 AIR 응용 프로그램을 지원하는 첫 번째 버전의 AIR인 TV용 AIR 2.5를 지원합니다.

- Adobe Flash® Builder®

  Flash Builder 4.5는 TV용 AIR 2.5를 지원합니다.

- AIR SDK

  AIR 2.5부터는 AIR SDK와 함께 제공되는 명령줄 도구를 사용하여 응용 프로그램을 개발할 수 있습니다. AIR SDK를 다운로드하려면 http://www.adobe.com/kr/products/air/sdk/를 참조하십시오.

Flash Professional 사용

Flash Professional을 사용하여 TV용 AIR 응용 프로그램을 개발, 테스트 및 제작하는 것은 AIR 데스크톱 응용 프로그램의 도구를 사용하는 것과 비슷합니다.

하지만 ActionScript 3.0 코드를 작성할 때는 tv 및 extendedTV AIR 프로파일이 지원하는 클래스와 메서드만 사용하십시오. 자세한 내용은 222페이지의 "장치 프로파일"을 참조하십시오.

프로젝트 설정

TV용 AIR 응용 프로그램의 프로젝트를 설정하려면 다음을 수행하십시오.

- [제작 설정] 대화 상자의 [Flash] 탭에서 [플레이어] 값을 최소한 [AIR 2.5]로 설정합니다.
- [Adobe AIR 설정] 대화 상자(응용 프로그램 및 설치 프로그램 설정)의 [일반] 탭에서 프로파일을 TV 또는 확장 TV로 설정합니다.

디버깅

Flash Professional 내에서 AIR Debug Launcher를 사용하여 응용 프로그램을 실행할 수 있습니다. 다음을 수행하십시오.

- 디버깅 모드에서 응용 프로그램을 실행하려면 다음을 선택합니다.

  디버그 > 동영상 디버그 > AIR Debug Launcher(데스크톱)

  이렇게 선택한 다음 이후의 디버깅 실행에 대해서는 다음을 선택하면 됩니다.

  디버그 > 동영상 디버그 > 디버그

- 디버깅 모드 기능 없이 응용 프로그램을 실행하려면 다음을 선택합니다.

  컨트롤 > 동영상 테스트 > AIR Debug Launcher(데스크톱)

  이렇게 선택한 다음 이후 실행에 대해서는 [컨트롤] > [동영상 테스트] > [테스트]를 선택하면 됩니다.

AIR 프로파일이 TV 또는 확장 TV로 설정되기 전에 AIR Debug Launcher는 [리모컨 버튼]이라는 메뉴를 제공합니다. 이 메뉴를 사용하여 리모컨 장치에서 키를 누르는 것과 같은 효과를 낼 수 있습니다.

자세한 내용은 132페이지의 "Flash Professional을 사용한 원격 디버깅"을 참조하십시오.

기본 확장 사용

응용 프로그램에서 기본 확장을 사용할 경우 136페이지의 "기본 확장 사용을 위한 작업 목록"의 지침을 따르십시오.

하지만 응용 프로그램에서 기본 확장을 사용하는 경우에는 다음과 같습니다.

- Flash Professional을 사용하여 응용 프로그램을 제작할 수 없습니다. 응용 프로그램을 제작하려면 ADT를 사용하십시오. 128페이지의 "ADT를 사용하여 패키지화"를 참조하십시오.
- Flash Professional을 사용하여 응용 프로그램을 실행 또는 디버깅할 수 없습니다. 개발 컴퓨터에서 응용 프로그램을 디버깅하려면 ADL을 사용하십시오. 130페이지의 "ADL을 사용하는 장치 시뮬레이션"을 참조하십시오.

Flash Builder 사용

Flash Builder 4.5부터는 Flash Builder에서 TV용 AIR 개발을 지원합니다. Flash Builder를 사용하여 TV용 AIR 응용 프로그램을 개발, 테스트 및 제작하는 것은 AIR 데스크톱 응용 프로그램의 도구를 사용하는 것과 비슷합니다.

응용 프로그램 설정

응용 프로그램이 다음 조건을 충족하는지 확인하십시오.

- MXML 파일을 사용하는 경우 MXML 파일에서 Application 요소를 컨테이너 클래스로 사용합니다.

```
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
    xmlns:s="library://ns.adobe.com/flex/spark"
    xmlns:mx="library://ns.adobe.com/flex/mx">

    <!-- Place elements here. -->

</s:Application>.
```

중요: TV용 AIR 응용 프로그램은 WindowedApplication 요소를 지원하지 않습니다.

참고: 따라서 MXML 파일을 전혀 사용할 필요가 없습니다. 대신 ActionScript 3.0 프로젝트를 만들 수 있습니다.

- tv 및 extendedTV AIR 프로파일이 지원하는 ActionScript 3.0 클래스 및 메서드만 사용합니다. 자세한 내용은 222페이지의 “장치 프로파일”을 참조하십시오.

또한 응용 프로그램의 XML 파일에서 다음 조건을 충족하는지 확인하십시오.

- application 요소의 xmlns 특성이 AIR 2.5로 설정되어 있습니다.

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

- supportedProfiles 요소에 tv 또는 extendedTV가 포함되어 있습니다.

```
<supportedProfiles>tv</supportedProfiles>
```

응용 프로그램 디버깅

Flash Builder 내에서 AIR Debug Launcher를 사용하여 응용 프로그램을 실행할 수 있습니다. 다음을 수행하십시오.

1. [실행] > [디버그 구성]을 선택합니다.
2. [프로파일] 필드가 [데스크톱]으로 설정되어 있는지 확인합니다.
3. [실행] > [디버그]를 선택하여 디버깅 모드로 실행하거나, [실행] > [실행]을 선택하여 디버깅 모드 기능 없이 실행합니다.

supportedProfiles 요소가 TV 또는 확장 TV로 설정되어 있기 때문에 AIR Debug Launcher는 [리모컨 버튼]이라는 메뉴를 제공합니다. 이 메뉴를 사용하여 리모컨 장치에서 키를 누르는 것과 같은 효과를 낼 수 있습니다.

자세한 내용은 132페이지의 “Flash Builder를 사용한 원격 디버깅”을 참조하십시오.

기본 확장 사용

응용 프로그램에서 기본 확장을 사용할 경우 136페이지의 “기본 확장 사용을 위한 작업 목록”의 지침을 따르십시오.

하지만 응용 프로그램에서 기본 확장을 사용하는 경우에는 다음과 같습니다.

- Flash Builder를 사용하여 응용 프로그램을 제작할 수는 없습니다. 응용 프로그램을 제작하려면 ADT를 사용하십시오. 128페이지의 “ADT를 사용하여 패키지화”를 참조하십시오.
- Flash Builder를 사용하여 응용 프로그램을 실행 또는 디버깅할 수 없습니다. 개발 컴퓨터에서 응용 프로그램을 디버깅하려면 ADL을 사용하십시오. 130페이지의 “ADL을 사용하는 장치 시뮬레이션”을 참조하십시오.

TV용 AIR 응용 프로그램 설명자 속성

다른 AIR 응용 프로그램에서와 마찬가지로 응용 프로그램 설명자 파일에서 기본 응용 프로그램 속성을 설정합니다. TV 프로파일 응용 프로그램은 윈도우 크기, 투명도 등 일부 데스크톱 관련 속성을 무시합니다. extendedTV 프로파일에서 장치를 대상으로 하는 응용 프로그램은 기본 확장을 사용할 수 있습니다. 이러한 응용 프로그램은 extensions 요소에서 사용된 기본 확장을 식별합니다.

공통 설정

모든 TV 프로파일 응용 프로그램에서 중요한 여러 가지 응용 프로그램 설명자 설정이 있습니다.

필요한 AIR 런타임 버전

응용 프로그램 설명자 파일의 네임스페이스를 사용하여 응용 프로그램에 필요한 AIR 런타임의 버전을 지정합니다.

응용 프로그램에서 사용할 수 있는 기능을 결정하는 데 가장 큰 영향을 미치는 것은 application 요소에 할당된 네임스페이스입니다. 예를 들어 AIR 2.5 네임스페이스를 사용하지만 사용자가 특정 이후 버전을 설치한 응용 프로그램을 가정해 보십시오. 이 경우 이후의 AIR 버전에서 비헤이비어가 다르지만 응용 프로그램은 여전히 AIR 2.5 비헤이비어를 인식합니다. 네임스페이스를 변경하고 업데이트를 제작할 경우에만 응용 프로그램이 새로운 비헤이비어 및 기능에 액세스할 수 있습니다. 보안 픽스는 이 규칙에 대한 중요한 예외입니다.

루트 application 요소의 xmlns 특성을 사용하여 네임스페이스를 지정합니다.

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

AIR 2.5는 TV 응용 프로그램을 지원하는 최초의 AIR 버전입니다.

응용 프로그램 ID

제작하는 각 응용 프로그램에 대해 여러 설정이 고유해야 합니다. 이러한 설정에는 id, name 및 filename 요소가 포함됩니다.

```
<id>com.example.MyApp</id>
<name>My Application</name>
<filename>MyApplication</filename>
```

응용 프로그램 버전

versionNumber 요소에서 응용 프로그램 버전을 지정합니다. versionNumber 값을 지정할 때는 최대 세 개의 숫자로 이루어진 시퀀스를 사용할 수 있으며, 이때 각 숫자는 점으로 분리해야 합니다(예: “0.1.2”). 버전 번호의 각 세그먼트는 최대 세 자리까지 가능합니다. 즉, “999.999.999”가 허용되는 가장 큰 버전 번호입니다. 번호에 세 세그먼트를 모두 포함해야 하는 것은 아닙니다. 따라서 “1” 및 “1.0”도 유효한 버전 번호입니다.

versionLabel 요소를 사용하여 버전에 대한 레이블을 지정할 수도 있습니다. 버전 레이블을 추가하면 버전 번호 대신 표시됩니다.

```
<versionNumber>1.23.7<versionNumber>
<versionLabel>1.23 Beta 7</versionLabel>
```

기본 응용 프로그램 SWF

initialWindow 요소의 versionLabel 자식에서 기본 응용 프로그램 SWF 파일을 지정합니다. TV 프로파일에 있는 장치를 대상으로 삼을 때는 SWF 파일을 사용해야 합니다(HTML 기반 응용 프로그램은 지원되지 않음).

```
<initialWindow>
    <content>MyApplication.swf</content>
</initialWindow>
```

파일을 AIR 패키지에 포함해야 합니다(ADT 또는 IDE 사용). 단순히 응용 프로그램 설명자에 있는 이름을 참조하는 것만으로는 파일이 패키지에 자동으로 포함되지 않습니다.

기본 화면 속성

initialWindow 요소의 여러 자식 요소는 기본 응용 프로그램 화면의 초기 모양 및 비헤이비어를 제어합니다. 이들 속성 중 대부분은 TV 프로파일에 있는 장치에서 무시되지만 fullScreen 요소는 사용할 수 있습니다.

- fullScreen - 응용 프로그램이 전체 장치 디스플레이를 차지해야 하는지 아니면 일반 운영 체제 크롬과 디스플레이를 공유해야 하는지를 지정합니다.

  ```
  <fullScreen>true</fullScreen>
  ```

visible 요소

visible 요소는 initialWindow 요소의 자식 요소입니다. TV용 AIR 장치에는 응용 프로그램의 내용이 항상 표시되기 때문에 TV용 AIR에서는 visible 요소를 무시합니다.

하지만 응용 프로그램에서 데스크톱 장치를 대상으로 지정한 경우에는 visible 요소를 true로 설정합니다.

데스크톱 장치에서는 이 요소의 값이 기본적으로 false로 지정됩니다. 따라서 visible 요소를 포함하지 않으면 데스크톱 장치에서 응용 프로그램의 내용이 표시되지 않습니다. ActionScript 클래스인 NativeWindow를 사용하여 데스크톱 장치에 내용을 표시할 수 있지만 TV 장치 프로파일에서는 NativeWindow 클래스를 지원하지 않습니다. 따라서 TV용 AIR 장치에서 실행되는 응용 프로그램에서 NativeWindow 클래스를 사용하려고 하면 응용 프로그램이 로드되지 않습니다. NativeWindow 클래스의 메서드를 호출하는지 여부와 관계없이 이 클래스를 사용하는 응용 프로그램은 TV용 AIR 장치에 로드되지 않습니다.

지원되는 프로파일

응용 프로그램이 TV 장치에서만 작동하는 경우에는 다른 유형의 컴퓨팅 장치에 설치되는 것을 방지할 수 있습니다. supportedProfiles 요소의 지원되는 목록에서 다른 프로파일을 제외합니다.

```
<supportedProfiles>tv extendedTV</supportedProfiles>
```

응용 프로그램에서 기본 확장을 사용하는 경우 지원되는 프로파일 목록에 extendedTV 프로파일만 포함합니다.

```
<supportedProfiles>extendedTV</supportedProfiles>
```

supportedProfiles 요소를 생략하면 응용 프로그램이 모든 프로파일을 지원하는 것으로 가정합니다.

supportedProfiles 목록에 tv 프로파일만 포함해서는 안 됩니다. 일부 TV 장치에서는 TV용 AIR를 항상 extendedTV 프로파일에 해당하는 모드로 실행합니다. 이 비헤이비어는 TV용 AIR가 기본 확장을 사용하는 응용 프로그램을 실행할 수 있도록 하기 위한 것입니다. supportedProfiles 요소가 tv만 지정하는 경우에는 내용이 extendedTV에 대한 TV용 AIR 모드와 호환되지 않음을 알리는 것입니다. 따라서 일부 TV 장치에서는 tv 프로파일만 지정하는 응용 프로그램을 로드하지 않습니다.

tv 및 extendedTV 프로파일에서 지원되는 ActionScript 클래스의 목록에 대해서는 223페이지의 “각 프로파일의 기능”을 참조하십시오.

필수 기본 확장

extendedTV 프로파일을 지원하는 응용 프로그램에서는 기본 확장을 사용할 수 있습니다.

extensions 및 extensionID 요소를 사용하여 AIR 응용 프로그램이 응용 프로그램 설명자에서 사용하는 모든 기본 확장을 선언합니다. 다음 예제에서는 두 개의 필수 기본 확장을 지정하기 위한 구문을 보여 줍니다.

```
<extensions>
    <extensionID>com.example.extendedFeature</extensionID>
    <extensionID>com.example.anotherFeature</extensionID>
</extensions>
```

확장이 목록에 없는 경우에는 응용 프로그램에서 해당 확장을 사용할 수 없습니다.

extensionID 요소의 값은 확장 설명자 파일에서 id 요소의 값과 동일합니다. 확장 설명자 파일은 extension.xml이라는 XML 파일로, 장치 제조업체로부터 받는 ANE 파일에 패키지화되어 있습니다.

extensions 요소에 확장을 지정하지만 TV용 AIR 장치에 확장이 설치되어 있지 않으면 응용 프로그램을 실행할 수 없습니다. TV용 AIR 응용 프로그램과 패키지화하는 ANE 파일에 스텁 버전의 확장이 있는 경우에는 이러한 규칙이 적용되지 않습니다. 이 경우 응용 프로그램을 실행할 수 있고 응용 프로그램에서 스텁 버전의 확장을 사용합니다. 스텁 버전에는 ActionScript 코드는 있지만 기본 코드는 없습니다.

응용 프로그램 아이콘

TV 장치에서 응용 프로그램 아이콘에 대한 요구 사항은 장치마다 다릅니다. 예를 들어 장치 제조업체는 다음을 지정합니다.

- 필수 아이콘 및 아이콘 크기

- 필수 파일 유형 및 명명 규칙
- 아이콘을 응용 프로그램과 함께 패키지화하는지 여부 등 응용 프로그램의 아이콘을 제공하는 방법
- 응용 프로그램 설명자 파일의 icon 요소에서 아이콘을 지정하는지 여부
- 응용 프로그램에서 아이콘을 제공하지 않을 경우의 비헤이비어

자세한 내용은 장치 제조업체에 문의하십시오.

무시되는 설정

TV 장치의 응용 프로그램은 모바일, 기본 윈도우 또는 데스크톱 운영 체제 기능에 적용되는 응용 프로그램 설정을 무시합니다. 무시되는 설정은 다음과 같습니다.

- allowBrowserInvocation
- aspectRatio
- autoOrients
- customUpdateUI
- fileTypes
- height
- installFolder
- maximizable
- maxSize
- minimizable
- minSize
- programMenuFolder
- renderMode
- resizable
- systemChrome
- title
- transparent
- visible
- width
- x
- y

TV용 AIR 응용 프로그램 패키지화

ADT를 사용하여 패키지화

AIR ADT 명령줄 도구를 사용하여 TV용 AIR 응용 프로그램을 패키지화할 수 있습니다. AIR SDK 버전 2.5부터 ADT는 TV 장치에 대해 패키지화를 지원합니다. 패키지화하기 전에 모든 ActionScript 및 MXML 코드를 컴파일하십시오. 코드 서명 인증서도 있어야 합니다. 인증서는 ADT -certificate 명령을 사용하여 만들 수 있습니다.

ADT 명령 및 옵션에 대한 자세한 참조는 149페이지의 “ADT(AIR Developer Tool)”를 참조하십시오.

AIR 패키지 만들기

AIR 패키지를 만들려면 ADT package 명령을 사용하십시오.

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 myApp.air myApp-app.xml myApp.swf icons
```

이 예제에서는 다음을 가정합니다.

- ADT 도구의 경로가 명령줄 셸의 경로 정의에 있습니다. 277페이지의 “path 환경 변수”를 참조하십시오.
- 인증서 codesign.p12가 현재 ADT 명령을 실행하고 있는 상위 디렉토리에 있습니다.

응용 프로그램 파일이 들어 있는 디렉토리에서 명령을 실행합니다. 이 예제의 응용 프로그램 파일은 myApp-app.xml(응용 프로그램 설명자 파일), myApp.swf 및 icons 디렉토리입니다.

표시된 것과 같은 명령을 실행하면 ADT에서 키 저장소 암호를 묻습니다. 입력하는 암호 문자가 모든 셸 프로그램에서 표시되는 것은 아닙니다. 입력을 마쳤으면 Enter를 누르면 됩니다. 또는 storepass 매개 변수를 사용하여 ADT 명령에 암호를 포함할 수 있습니다.

AIRN 패키지 만들기

TV용 AIR 응용 프로그램에서 기본 확장을 사용하는 경우 AIR 패키지 대신 AIRN 패키지를 만듭니다. AIRN 패키지를 만들려면 ADT package 명령을 사용하여 대상 유형을 airn으로 설정하십시오.

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp-app.xml myApp.swf
icons -extdir C:\extensions
```

이 예제에서는 다음을 가정합니다.

- ADT 도구의 경로가 명령줄 셸의 경로 정의에 있습니다. 277페이지의 “path 환경 변수”를 참조하십시오.
- 인증서 codesign.p12가 현재 ADT 명령을 실행하고 있는 상위 디렉토리에 있습니다.
- 매개 변수 -extdir은 응용 프로그램에서 사용하는 ANE 파일이 포함된 디렉토리의 이름을 지정합니다.

  이러한 ANE 파일에는 전용 스텁 또는 시뮬레이션 버전의 ActionScript 확장이 포함되어 있습니다. TV용 AIR 장치에는 기본 코드를 포함하고 있는 확장 버전이 설치되어 있습니다.

응용 프로그램 파일이 들어 있는 디렉토리에서 명령을 실행합니다. 이 예제의 응용 프로그램 파일은 myApp-app.xml(응용 프로그램 설명자 파일), myApp.swf 및 icons 디렉토리입니다.

표시된 것과 같은 명령을 실행하면 ADT에서 키 저장소 암호를 묻습니다. 입력하는 암호 문자가 모든 셸 프로그램에서 표시되는 것은 아닙니다. 입력을 마쳤으면 Enter를 누르면 됩니다. 또는 storepass 매개 변수를 사용하여 명령에 암호를 포함할 수 있습니다.

기본 확장을 사용하는 TV용 AIR 응용 프로그램에 대한 AIRI 파일을 만들 수도 있습니다. AIRI 파일은 서명되지 않은 점을 제외하고는 AIRN 파일과 똑같습니다. 예를 들면 다음과 같습니다.

```
adt -prepare myApp.airi myApp.xml myApp.swf icons -extdir C:\extensions
```

그런 다음 응용 프로그램에 서명할 준비가 되면 AIRI 파일로부터 AIRN 파일을 만들 수 있습니다.

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp.airi
```

자세한 내용은 Adobe AIR용 기본 확장 개발을 참조하십시오.

Flash Builder 또는 Flash Professional을 사용하여 패키지화

Flash Professional 및 Flash Builder를 사용하면 ADT를 직접 실행하지 않고도 AIR 패키지를 제작하고 내보낼 수 있습니다. AIR 응용 프로그램을 위한 AIR 패키지를 만드는 절차는 해당 프로그램의 설명서에 나와 있습니다.

그러나 현재는 기본 확장을 사용하는 TV용 AIR 응용 프로그램을 위한 응용 프로그램 패키지인 AIRN 패키지를 ADT에서만 만들 수 있습니다.

자세한 내용은 다음을 참조하십시오.

- Flash Builder를 사용하여 AIR 응용 프로그램 패키지화
- Flash Professional을 사용하여 Adobe AIR용으로 제작

TV용 AIR 응용 프로그램 디버깅

ADL을 사용하는 장치 시뮬레이션

대부분의 응용 프로그램 기능을 가장 빠르고 쉽게 테스트하고 디버깅하는 방법은 ADL(Adobe Debug Launcher) 유틸리티를 사용하는 개발 컴퓨터에서 응용 프로그램을 실행하는 것입니다.

ADL은 응용 프로그램 설명자에서 supportedProfiles 요소를 사용하여 사용할 프로파일을 선택합니다. 특히 다음 값에 유의하십시오.

- 목록에 여러 개의 프로파일이 있는 경우 ADL은 목록에서 첫 번째 프로파일을 사용합니다.
- ADL의 -profile 매개 변수를 사용하여 supportedProfiles 목록에서 다른 프로파일 중 하나를 선택할 수 있습니다.
- 응용 프로그램 설명자에 supportedProfiles 요소를 포함하지 않는 경우에는 -profile 인수에 대해 아무 프로파일이나 지정하면 됩니다.

예를 들어 다음 명령을 사용하여 응용 프로그램을 시작함으로써 tv 프로파일을 시뮬레이트하십시오.

```
adl -profile tv myApp-app.xml
```

ADL을 사용하여 데스크톱에서 tv 또는 extendedTV 프로파일을 시뮬레이트할 때는 대상 장치와 좀 더 비슷한 환경에서 응용 프로그램이 실행됩니다. 예를 들면 다음과 같습니다.

- -profile 인수에 있는 프로파일에 속하지 않은 ActionScript API는 사용할 수 없습니다.
- ADL에서는 메뉴 명령을 통해 리모컨 등 장치 입력 컨트롤의 입력을 허용합니다.
- -profile 인수에서 tv 또는 extendedTV를 지정하면 ADL이 데스크톱에서 StageVideo 클래스를 시뮬레이트할 수 있습니다.
- -profile 인수에서 extendedTV를 지정하면 응용 프로그램이 응용 프로그램 AIRN 파일과 함께 패키지화된 기본 확장 스텁 또는 시뮬레이터를 사용할 수 있습니다.

하지만 ADL은 데스크톱에서 응용 프로그램을 실행하기 때문에 ADL을 사용하여 TV용 AIR 응용 프로그램을 테스트하는 데는 제한이 있습니다.

- 장치의 응용 프로그램 성능이 반영되지 않습니다. 대상 장치에서 성능 테스트를 실행합니다.
- StageVideo 객체의 제한을 시뮬레이트하지 않습니다. 일반적으로 TV용 AIR 장치를 대상으로 삼을 때는 Video 클래스가 아니라 StageVideo 클래스를 사용하여 비디오를 재생합니다. StageVideo 클래스는 장치 하드웨어의 성능 이점을 활용하지만 표시 제한이 있습니다. ADL은 이러한 제한 없이 데스크톱에서 비디오를 재생합니다. 따라서 대상 장치에서 재생 비디오를 테스트하십시오.
- 기본 확장의 기본 코드를 시뮬레이트할 수 없습니다. 하지만 ADL -profile 인수에서 기본 확장을 지원하는 extendedTV 프로파일을 지정할 수 있습니다. ADL에서는 ANE 패키지에 포함된 전용 스텁 또는 시뮬레이터 버전의 ActionScript 확장을 사용하여 테스트할 수 있습니다. 하지만 일반적으로 장치에 설치된 해당 확장에는 기본 코드도 포함되어 있습니다. 확장을 기본 코드와 함께 사용하여 테스트하려면 대상 장치에서 응용 프로그램을 실행하십시오.

자세한 내용은 144페이지의 “ADL(AIR Debug Launcher)”을 참조하십시오.

기본 확장 사용

응용 프로그램에서 기본 확장을 사용하는 경우 ADL 명령은 다음 예제처럼 표시됩니다.

```
adl -profile extendedTV -extdir C:\extensionDirs myApp-app.xml
```

이 예제에서는 다음을 가정합니다.

- ADL 도구의 경로가 명령줄 셸의 경로 정의에 있습니다. 277페이지의 “path 환경 변수”를 참조하십시오.
- 현재 디렉토리에 응용 프로그램 파일이 들어 있습니다. 이러한 파일에는 SWF 파일 및 응용 프로그램 설명자 파일(이 예제의 경우 myApp-app.xml)이 포함됩니다.
- 매개 변수 -extdir은 응용 프로그램에서 사용하는 각 기본 확장의 디렉토리를 포함하고 있는 디렉토리의 이름을 지정합니다. 이러한 각 디렉토리에는 기본 확장의 **패키지화되지 않은** ANE 파일이 포함되어 있습니다. 예를 들면 다음과 같습니다.

```
C:\extensionDirs
    extension1.ane
        META-INF
            ANE
                default
                    library.swf
                extension.xml
            signatures.xml
        catalog.xml
        library.swf
        mimetype
    extension2.ane
        META-INF
            ANE
                default
                    library.swf
                extension.xml
            signatures.xml
        catalog.xml
        library.swf
        mimetype
```

  이러한 패키지화되지 않은 ANE 파일에는 ActionScript 전용 스텁 또는 시뮬레이터 버전의 확장이 포함되어 있습니다. TV용 AIR 장치에는 기본 코드를 포함하고 있는 확장 버전이 설치되어 있습니다.

자세한 내용은 Adobe AIR용 기본 확장 개발을 참조하십시오.

컨트롤 입력

ADL은 TV 장치의 원격 컨트롤 버튼을 시뮬레이트합니다. TV 프로파일 중 하나를 사용하여 ADL을 실행할 때 표시되는 메뉴를 사용하여 시뮬레이트된 장치로 이러한 버튼 입력을 보낼 수 있습니다.

스크린 크기

ADL -screensize 매개 변수를 설정하여 다양한 크기의 스크린에서 응용 프로그램을 테스트할 수 있습니다. 일반 스크린 및 최대화된 스크린의 폭과 높이를 나타내는 네 개의 값이 포함된 문자열을 지정할 수 있습니다.

항상 세로 레이아웃에 대한 픽셀 크기를 지정합니다. 이는 높이 값보다 작은 값을 폭에 지정해야 한다는 것을 의미합니다. 예를 들면 다음과 같습니다.

```
adl -screensize 728x1024:768x1024 myApp-app.xml
```

trace 문

데스크톱에서 TV 응용 프로그램을 실행하면 ADL을 실행하는 데 사용되는 디버거 또는 터미널 윈도우에 trace 출력이 인쇄됩니다.

Flash Professional을 사용한 원격 디버깅

TV용 AIR 응용 프로그램이 대상 장치에서 실행되는 동안 Flash Professional을 사용하여 TV용 AIR 응용 프로그램을 원격으로 디버깅할 수 있습니다. 하지만 원격 디버깅을 설정하는 단계는 장치마다 다릅니다. 예를 들어 Adobe® AIR® for TV MAX 2010 Hardware Development Kit에는 해당 장치에 대한 자세한 단계를 보여 주는 설명서가 포함되어 있습니다.

하지만 대상 장치에 관계없이 다음 단계를 수행하여 원격 디버깅을 준비하십시오.

1 [제작 설정] 대화 상자의 [Flash] 탭에서 [디버깅 허용]을 선택합니다.

이 옵션은 Flash Professional이 FLA 파일로부터 만드는 모든 SWF 파일에 디버깅 정보를 포함하도록 합니다.

2 [Adobe AIR 설정] 대화 상자(응용 프로그램 및 설치 프로그램 설정)의 [서명] 탭에서 AIRI(AIR Intermediate) 파일을 준비하는 옵션을 선택합니다.

여전히 응용 프로그램을 개발하는 중인 경우 디지털 서명이 불필요한 AIRI 파일을 사용하는 것으로 충분합니다.

3 응용 프로그램을 제작하고, AIRI 파일을 만듭니다.

마지막 단계에서는 대상 장치에서 응용 프로그램을 설치하고 실행합니다. 하지만 이러한 단계는 장치마다 다릅니다.

Flash Builder를 사용한 원격 디버깅

TV용 AIR 응용 프로그램이 대상 장치에서 실행되는 동안 Flash Builder를 사용하여 TV용 AIR 응용 프로그램을 원격으로 디버깅할 수도 있습니다. 하지만 원격 디버깅을 수행하는 단계는 장치마다 다릅니다.

하지만 대상 장치에 관계없이 다음 단계를 수행하여 원격 디버깅을 준비하십시오.

1 [프로젝트] > [릴리스 빌드 내보내기]를 선택합니다. AIRI(AIR Intermediate) 파일을 준비하는 옵션을 선택합니다.

여전히 응용 프로그램을 개발하는 중인 경우 디지털 서명이 불필요한 AIRI 파일을 사용하는 것으로 충분합니다.

2 응용 프로그램을 제작하고, AIRI 파일을 만듭니다.

3 디버그 정보가 들어 있는 SWF 파일을 포함하도록 응용 프로그램의 AIRI 패키지를 변경합니다.

디버그 정보가 들어 있는 SWF 파일은 bin-debug라는 디렉토리에서 응용 프로그램에 대한 Flash Builder 프로젝트 디렉토리에 있습니다. AIRI 패키지에 있는 SWF 파일을 bin-debug 디렉토리에 있는 SWF 파일과 바꿉니다.

Windows 개발 컴퓨터에서 다음을 수행하여 이 두 파일을 바꿀 수 있습니다.

1 AIRI 패키지 파일의 이름을 변경하여 파일 이름 확장자가 .airi 대신 .zip이 되도록 합니다.

2 ZIP 파일 내용을 추출합니다.

3 추출한 디렉토리 구조에 있는 SWF 파일을 bin-debug에 있는 SWF 파일과 바꿉니다.

4 추출한 디렉토리의 파일을 zip 파일로 다시 압축합니다.

5 압축된 파일을 다시 변경하여 파일 이름 확장자가 .airi가 되도록 합니다.

Mac 개발 컴퓨터를 사용 중인 경우 이와 같이 바꾸는 단계는 장치마다 다릅니다. 하지만 일반적으로 다음과 같은 과정이 포함됩니다.

1 대상 장치에 AIRI 패키지를 설치합니다.

2 대상 장치의 응용 프로그램 설치 디렉토리에 있는 SWF 파일을 bin-debug 디렉토리에 있는 SWF 파일과 바꿉니다.

Adobe AIR for TV MAX 2010 Hardware Development Kit에 포함되어 있는 장치를 예로 들어 보겠습니다. 키트 설명서에 설명된 대로 AIRI 패키지를 설치합니다. 그런 다음 Mac 개발 컴퓨터의 명령줄에서 telnet을 사용하여 대상 장치에 액세스합니다. 응용 프로그램 설치 디렉토리(/opt/adobe/stagecraft/apps/**<application name>**/)에 있는 SWF 파일을 bin-debug 디렉토리에 있는 SWF 파일과 바꿉니다.

다음 단계는 Flash Builder 그리고 Adobe AIR for TV MAX 2010 Hardware Development Kit에 포함되어 있는 장치를 사용하는 원격 디버깅의 경우에 적용됩니다.

1 Flash Builder를 실행하는 컴퓨터, 즉 개발 컴퓨터에서 MAX 2010 Hardware Development Kit와 함께 제공된 AIR for TV Device Connector를 실행합니다. 그러면 개발 컴퓨터의 IP 주소가 표시됩니다.

2 하드웨어 키트 장치에서 DevMaster 응용 프로그램을 시작합니다. 이 응용 프로그램도 개발 키트와 함께 제공됩니다.

3 DevMaster 응용 프로그램에서 AIR for TV Device Connector에 표시된 개발 컴퓨터의 IP 주소를 입력합니다.

4 DevMaster 응용 프로그램에서 [원격 디버깅 사용]이 선택되어 있는지 확인합니다.

5 DevMaster 응용 프로그램을 종료합니다.

6 개발 컴퓨터의 AIR for TV Connector에서 [시작]을 선택합니다.

7 하드웨어 키트 장치에서 다른 응용 프로그램을 시작합니다. AIR for TV Device Connector에서 추적 정보가 표시되는지 확인합니다.

   추적 정보가 표시되지 않으면 개발 컴퓨터 및 하드웨어 키트 장치가 연결되지 않은 것입니다. 추적 정보에 사용되는 개발 컴퓨터의 포트가 사용 가능한지 확인합니다. AIR for TV Device Connector에서 다른 포트를 선택할 수 있습니다. 또한 선택한 포트에 액세스하는 것을 방화벽에서 허용하는지 확인합니다.

그 다음에 Flash Builder에서 디버거를 시작합니다. 다음을 수행하십시오.

1 Flash Builder에서 [실행] > [디버그 구성]을 선택합니다.

2 로컬 디버깅을 위한 기존 디버그 구성에서 프로젝트의 이름을 복사합니다.

3 [디버그 구성] 대화 상자에서 [웹 응용 프로그램]을 선택합니다. 그런 다음 [새로운 시작 구성] 아이콘을 선택합니다.

4 [프로젝트] 필드에 프로젝트 이름을 붙여 넣습니다.

5 [시작할 URL 또는 경로] 섹션에서 [기본값 사용]의 선택을 취소합니다. 또한 텍스트 필드에 **about:blank**를 입력합니다.

6 [적용]을 선택하여 변경 내용을 저장합니다.

7 [디버그]를 선택하여 Flash Builder 디버거를 시작합니다.

8 하드웨어 키트 장치에서 응용 프로그램을 시작합니다.

이제 Flash Builder 디버거를 사용하여 중단점을 설정하거나 변수를 조사하는 등의 작업을 수행할 수 있습니다.

9장: Adobe AIR용 기본 확장 사용

Adobe AIR용 기본 확장은 기본 코드로 프로그래밍된 장치별 기능에 액세스할 수 있도록 하는 ActionScript API를 제공합니다. 기본 확장 개발자는 종종 장치 제조업체와 협력하며, 경우에 따라서는 타사 개발자이기도 합니다.

기본 확장을 개발하는 중이면 Adobe AIR용 기본 확장 개발을 참조하십시오.

기본 확장은 다음의 조합입니다.

- ActionScript 클래스
- 기본 코드

그러나 기본 확장을 사용하는 AIR 응용 프로그램 개발자의 경우 ActionScript 클래스만 사용합니다.

기본 확장은 다음과 같은 상황에 유용합니다.

- 기본 코드 구현을 통해 플랫폼별 기능에 액세스할 수 있습니다. 이러한 플랫폼별 기능은 기본 제공 ActionScript 클래스에서 사용할 수 없고 응용 프로그램별 ActionScript 클래스에서 구현할 수 없습니다. 기본 코드 구현은 장치별 하드웨어 및 소프트웨어에 액세스할 수 있으므로 이러한 기능을 제공할 수 있습니다.
- 경우에 따라 기본 코드 구현이 ActionScript만 사용하는 구현보다 빠를 수 있습니다.
- 기본 코드 구현은 ActionScript에서 레거시 기본 코드에 액세스할 수 있도록 합니다.

Adobe 개발자 센터에는 기본 확장의 몇 가지 예가 마련되어 있습니다. 예를 들어 AIR 응용 프로그램이 Android의 진동 기능에 액세스할 수 있게 하는 기본 확장이 있습니다. Adobe AIR용 기본 확장을 참조하십시오.

ANE(AIR Native Extension) 파일

기본 확장 개발자는 기본 확장을 ANE 파일로 패키지화합니다. ANE 파일은 기본 확장에 필요한 라이브러리와 리소스가 포함된 아카이브 파일입니다.

일부 장치의 경우 ANE 파일은 기본 확장에 사용되는 기본 코드 라이브러리를 포함합니다. 그러나 다른 장치의 경우 기본 코드 라이브러리는 장치에 설치됩니다. 기본 확장에 특정 장치에 대한 기본 코드가 전혀 없는 경우도 있는데, 이 경우 ActionScript만으로 구현됩니다.

AIR 응용 프로그램 개발자는 ANE 파일을 다음과 같이 사용합니다.

- SWC 파일을 라이브러리 경로에 포함하는 것과 동일한 방법으로 ANE 파일을 응용 프로그램의 라이브러리 경로에 포함합니다. 이렇게 하면 응용 프로그램에서 확장의 ActionScript 클래스를 참조할 수 있습니다.

  참고: 응용 프로그램을 컴파일할 경우 ANE에 대한 동적 링크를 사용해야 합니다. Flash Builder를 사용하면 ActionScript Builder Path Properties 패널에서 External을 지정하고, 명령줄을 사용하면 -external-library-path를 지정합니다.
- AIR 응용 프로그램을 사용하여 ANE 파일을 패키지화합니다.

기본 확장과 NativeProcess ActionScript 클래스 비교

ActionScript 3.0에서는 NativeProcess 클래스를 제공합니다. 이 클래스를 사용하면 AIR 응용 프로그램이 호스트 운영 체제에서 기본 프로세스를 실행할 수 있습니다. 이 기능은 플랫폼별 기능과 라이브러리에 액세스할 수 있도록 하는 기본 확장과 비슷합니다. NativeProcess 클래스를 사용할지 아니면 기본 확장을 사용할지를 결정할 때는 다음 사항을 고려합니다.

- extendedDesktop AIR 프로파일만이 NativeProcess 클래스를 지원합니다. 따라서 AIR 프로파일 mobileDevice 및 extendedMobileDevice가 있는 응용 프로그램의 경우 기본 확장만 선택할 수 있습니다.
- 일반적으로 기본 확장 개발자는 다양한 플랫폼을 위한 기본 구현을 제공하지만 개발자가 제공하는 ActionScript API는 대개 플랫폼 간에 동일합니다. NativeProcess 클래스를 사용할 경우 기본 프로세스를 시작하기 위한 ActionScript 코드는 플랫폼 간에 다를 수 있습니다.
- NativeProcess 클래스는 별도의 프로세스를 시작하지만 기본 확장은 AIR 응용 프로그램과 동일한 프로세스에서 실행됩니다. 따라서 코드 충돌이 염려될 경우 NativeProcess 클래스를 사용하는 것이 더 안전합니다. 그러나 별도의 프로세스이므로 프로세스 간의 통신 처리를 구현해야 할 수 있습니다.

기본 확장과 ActionScript 클래스 라이브러리(SWC 파일) 비교

SWC 파일은 보관 형식의 ActionScript 클래스 라이브러리입니다. SWC 파일에는 SWC 파일 및 다른 리소스 파일이 포함되어 있습니다. SWC 파일은 개별 ActionScript 코드 및 리소스 파일을 공유하는 대신 ActionScript 클래스를 공유하는 편리한 수단으로 사용됩니다.

기본 확장 패키지는 ANE 파일입니다. SWC 파일과 마찬가지로 ANE 파일도 SWF 파일 및 기타 리소스 파일을 보관 형식으로 포함하는 ActionScript 클래스 라이브러리입니다. 그러나 ANE 파일과 SWC 파일 간의 가장 중요한 차이는 ANE 파일에만 기본 코드 라이브러리가 포함될 수 있다는 것입니다.

참고: 응용 프로그램을 컴파일할 경우 ANE 파일에 대한 동적 링크를 사용해야 합니다. Flash Builder를 사용하면 ActionScript Builder Path Properties 패널에서 External을 지정하고, 명령줄을 사용하면 -external-library-path를 지정합니다.

기타 도움말 항목

SWC 파일 정보

지원되는 장치

AIR 3부터는 다음 장치의 응용 프로그램에서 기본 확장을 사용할 수 있습니다.

- Android 장치(Android 2.2 이상)
- iOS 장치(iOS 4.0 이상)
- iOS 시뮬레이터(AIR 3.3 이상)
- Blackberry PlayBook
- AIR 3.0을 지원하는 Windows 데스크톱 장치
- AIR 3.0을 지원하는 Mac OS X 데스크톱 장치

일반적으로 동일한 기본 확장은 여러 플랫폼을 대상으로 합니다. 확장의 ANE 파일에는 지원되는 각 플랫폼에 대한 ActionScript 및 기본 라이브러리가 포함되어 있습니다. 일반적으로 ActionScript 라이브러리의 공용 인터페이스는 모든 플랫폼에 동일합니다. 기본 라이브러리는 물론 다릅니다.

경우에 따라 기본 확장은 기본 플랫폼을 지원합니다. 기본 플랫폼 구현에는 ActionScript 코드만 있고 기본 코드는 없습니다. 확장에서 특별히 지원하지 않는 플랫폼을 위한 응용 프로그램을 패키지화할 경우 해당 응용 프로그램은 실행될 때 기본 구현을 사용합니다. 예를 들어 모바일 장치에만 적용되는 기능을 제공하는 확장이 있다고 가정할 경우 이 확장에서는 데스크톱 응용 프로그램이 기능을 시뮬레이트하는 데 사용할 수 있는 기본 구현도 제공합니다.

지원되는 장치 프로파일

다음은 기본 확장을 지원하는 AIR 프로파일입니다.

- extendedDesktop(AIR 3.0 이상)
- mobileDevice(AIR 3.0 이상)
- extendedMobileDevice(AIR 3.0 이상)

기타 도움말 항목

AIR 프로파일 지원

기본 확장 사용을 위한 작업 목록

응용 프로그램에서 기본 확장을 사용하려면 다음 작업을 수행합니다.

1. 응용 프로그램 설명자 파일에서 확장을 선언합니다.
2. 응용 프로그램의 라이브러리 경로에 ANE 파일을 포함합니다.
3. 응용 프로그램을 패키지화합니다.

응용 프로그램 설명자 파일에서 확장 선언

모든 AIR 응용 프로그램에는 응용 프로그램 설명자 파일이 있습니다. 응용 프로그램에 기본 확장이 사용될 경우 응용 프로그램 설명자 파일에는 <extensions> 요소가 포함됩니다. 예를 들면 다음과 같습니다.

```
<extensions>
    <extensionID>com.example.Extension1</extensionID>
    <extensionID>com.example.Extension2</extensionID>
</extensions>
```

extensionID 요소의 값은 확장 설명자 파일에서 id 요소의 값과 동일합니다. 확장 설명자 파일은 extension.xml이라는 XML 파일로, 이 파일은 ANE 파일에 패키지화되어 있습니다. extension.xml 파일을 보려면 보관 추출기 도구를 사용하면 됩니다.

응용 프로그램의 라이브러리 경로에 ANE 파일 포함

기본 확장을 사용하는 응용 프로그램을 컴파일하려면 ANE 파일을 라이브러리 경로에 포함합니다.

Flash Builder와 함께 ANE 파일 사용

응용 프로그램에서 기본 확장을 사용하는 경우 라이브러리 경로에서 기본 확장에 대한 ANE 파일을 포함합니다. 그러면 Flash Builder를 사용하여 ActionScript 코드를 컴파일할 수 있습니다.

Flash Builder 4.5.1을 사용하는 다음 단계를 수행하십시오.

1 ANE 파일의 파일 이름 확장자를 .ane에서 .swc로 변경합니다. Flash Builder에서 파일을 찾을 수 있도록 하려면 이 단계를 수행해야 합니다.

2 Flash Builder 프로젝트에서 [프로젝트] > [속성]을 선택합니다.

3 [속성] 대화 상자에서 [Flex 빌드 경로]를 선택합니다.

4 [라이브러리 경로] 탭에서 [SWC 추가...]를 선택합니다.

5 SWC 파일을 찾아서 [열기]를 선택합니다.

6 [SWC 추가...] 대화 상자에서 [확인]을 선택합니다.

[속성] 대화 상자에서 [라이브러리 경로] 탭에 ANE 파일이 나타납니다.

7 SWC 파일 항목을 확장합니다. [링크 유형]을 두 번 클릭하여 [라이브러리 경로 항목 옵션] 대화 상자를 엽니다.

8 [라이브러리 경로 항목 옵션] 대화 상자에서 [링크 유형]을 [외부]로 변경합니다.

이제 [프로젝트] > [프로젝트 만들기] 등을 사용하여 응용 프로그램을 컴파일할 수 있습니다.

Flash Professional과 함께 ANE 파일 사용

응용 프로그램에서 기본 확장을 사용하는 경우 라이브러리 경로에서 기본 확장에 대한 ANE 파일을 포함합니다. 그러면 Flash Professional CS5.5를 사용하여 ActionScript 코드를 컴파일할 수 있습니다. 다음을 수행하십시오.

1 ANE 파일의 파일 이름 확장자를 .ane에서 .swc로 변경합니다. Flash Professional에서 파일을 찾을 수 있도록 하려면 이 단계를 수행해야 합니다.

2 FLA 파일에서 [파일] > [ActionScript 설정]을 선택합니다.

3 [고급 ActionScript 3.0 설정] 대화 상자에서 [라이브러리 경로] 탭을 선택합니다.

4 [SWC 파일 찾아보기] 버튼을 선택합니다.

5 SWC 파일을 찾아서 [열기]를 선택합니다.

[고급 ActionScript 3.0 설정] 대화 상자에서 [라이브러리 경로] 탭에 SWF 파일이 나타납니다.

6 SWC 파일이 선택된 상태에서 [라이브러리에 대한 링크 옵션 선택] 버튼을 선택합니다.

7 [라이브러리 경로 항목 옵션] 대화 상자에서 [링크 유형]을 [외부]로 변경합니다.

기본 확장을 사용하는 응용 프로그램 패키지화

기본 확장을 사용하는 응용 프로그램을 패키지화하려면 ADT를 사용합니다. Flash Professional CS5.5 또는 Flash Builder 4.5.1을 사용하여 응용 프로그램을 패키지화할 수는 없습니다.

ADT를 사용하는 방법에 대한 자세한 내용은 ADT(AIR Developer Tool)를 참조하십시오.

예를 들어 다음 ADT 명령을 실행하면 기본 확장을 사용하는 응용 프로그램에 대한 DMG 파일(Mac OS X용 기본 설치 프로그램 파일)이 생성됩니다.

```
adt -package
    -storetype pkcs12
    -keystore myCert.pfx
    -target native
    myApp.dmg
    application.xml
    index.html resources
    -extdir extensionsDir
```

다음 명령을 실행하면 Android 장치용 APK 패키지가 만들어집니다.

```
adt -package
    -target apk
    -storetype pkcs12 -keystore ../codesign.p12
    myApp.apk
    myApp-app.xml
    myApp.swf icons
    -extdir extensionsDir
```

다음 명령을 실행하면 iPhone 응용 프로그램용 iOS 패키지가 만들어집니다.

```
adt -package
    -target ipa-ad-hoc
    -storetype pkcs12 -keystore ../AppleDistribution.p12
    -provisioning-profile AppleDistribution.mobileprofile
    myApp.ipa
    myApp-app.xml
    myApp.swf icons Default.png
    -extdir extensionsDir
```

다음 사항에 주의하십시오.

- 기본 설치 프로그램 패키지 유형을 사용합니다.
- 확장 디렉토리를 지정합니다.
- ANE 파일이 응용 프로그램의 대상 장치를 지원하는지 확인합니다.

기본 설치 프로그램 패키지 유형 사용

응용 프로그램 패키지는 기본 설치 프로그램이어야 합니다. 기본 확장은 일반적으로 기본 코드를 포함하므로 기본 확장을 사용하는 응용 프로그램에 대해 플랫폼의 영향을 받지 않는 AIR 패키지(.air 패키지)를 만들 수는 없습니다. 그러나 일반적으로 기본 확장은 동일한 ActionScript API를 가진 여러 기본 플랫폼을 지원합니다. 이러한 경우 서로 다른 기본 설치 프로그램 패키지에서 동일한 ANE 파일을 사용할 수 있습니다.

다음 표에는 ADT 명령의 -target 옵션에 사용되는 값이 요약되어 있습니다.

응용 프로그램의 대상 플랫폼	**-target**
Mac OS X 또는 Windows 데스크톱 장치	-target native -target bundle
Android	-target apk 또는 기타 Android 패키지 대상
iOS	-target ipa-ad-hoc 또는 기타 iOS 패키지 대상
iOS 시뮬레이터	-target ipa-test-interpreter-simulator -target ipa-debug-interpreter-simulator

확장 디렉토리 지정

ADT 옵션인 -extdir은 기본 확장(ANE 파일)을 포함하는 디렉토리를 ADT에 알리는 데 사용합니다.

이 옵션에 대한 자세한 내용은 163페이지의 “파일 및 경로 옵션”을 참조하십시오.

ANE 파일이 응용 프로그램의 대상 장치를 지원하는지 확인

ANE 파일을 제공할 때 기본 확장 개발자는 확장에서 지원되는 플랫폼을 알려 줍니다. 또한 보관 추출기 도구를 사용하여 ANE 파일의 내용을 볼 수도 있습니다. 추출된 파일에는 지원되는 각 플랫폼에 대한 디렉토리가 포함되어 있습니다.

ANE 파일을 사용하는 응용 프로그램을 패키지화할 때는 확장에서 지원되는 플랫폼을 알고 있어야 합니다. 다음과 같은 규칙에 주의합니다.

- Android 응용 프로그램 패키지를 만들려면 ANE 파일에 Android-ARM 플랫폼이 포함되어 있어야 합니다. 아니면 ANE 파일에 기본 플랫폼과 하나 이상의 다른 플랫폼이 포함되어 있어야 합니다.
- iOS 응용 프로그램 패키지를 만들려면 ANE 파일에 iPhone-ARM 플랫폼이 포함되어 있어야 합니다. 아니면 ANE 파일에 기본 플랫폼과 하나 이상의 다른 플랫폼이 포함되어 있어야 합니다.
- iOS 시뮬레이터 응용 프로그램 패키지를 만들려면 ANE 파일에 iPhone-x86 플랫폼이 포함되어 있어야 합니다.
- Mac OS X 응용 프로그램 패키지를 만들려면 ANE 파일에 MacOS-x86 플랫폼이 포함되어 있어야 합니다. 아니면 ANE 파일에 기본 플랫폼과 하나 이상의 다른 플랫폼이 포함되어 있어야 합니다.
- Windows 응용 프로그램 패키지를 만들려면 ANE 파일에 Windows-x86 플랫폼이 포함되어 있어야 합니다. 아니면 ANE 파일에 기본 플랫폼과 하나 이상의 다른 플랫폼이 포함되어 있어야 합니다.

10장: ActionScript 컴파일러

ActionScript 및 MXML 코드를 AIR 응용 프로그램에 포함시키려면 먼저 컴파일해야 합니다. Adobe Flash Builder 또는 Adobe Flash Professional 같은 IDE(Integrated Development Environment)를 사용하는 경우에는 IDE가 컴파일을 백그라운드에서 처리합니다. 하지만 IDE를 사용하고 있지 않거나 빌드 스크립트를 사용할 때는 명령줄에서 ActionScript 컴파일러를 호출하여 SWF 파일을 만들 수 있습니다.

Flex SDK의 AIR 명령줄 도구

Adobe AIR 응용 프로그램을 만들기 위해 사용하는 각 명령줄 도구에서는 응용 프로그램을 만들기 위해 사용되는 해당 도구를 호출할 수 있습니다.

- amxmlc는 mxmlc를 호출하여 응용 프로그램 클래스를 컴파일합니다.
- acompc는 compc를 호출하여 라이브러리 및 구성 요소 클래스를 컴파일합니다.
- aasdoc는 asdoc를 호출하여 소스 코드 설명에서 문서 파일을 생성합니다.

이러한 유틸리티의 Flex 및 AIR 버전 간의 유일한 차이점은 AIR 버전의 경우 flex-config.xml 파일이 아닌 air-config.xml 파일에서 구성 옵션을 로드한다는 점입니다.

Flex SDK 도구 및 해당 명령줄 옵션은 Flex 설명서에서 자세히 설명합니다. 여기에서는 Flex SDK 도구에 대해 도구 사용을 시작하고 Flex 응용 프로그램 만들기와 AIR 응용 프로그램 만들기 사이의 차이점을 이해할 수 있도록 기본적인 내용이 설명됩니다.

기타 도움말 항목

34페이지의 “Flex SDK를 사용하여 첫 번째 데스크톱 AIR 응용 프로그램 만들기”

컴파일러 설정

일반적으로 명령줄 및 하나 이상의 구성 파일에서 컴파일 옵션을 지정합니다. 전역 Flex SDK 구성 파일에는 컴파일러가 실행될 때마다 사용되는 기본 값이 포함됩니다. 사용자의 고유 개발 환경에 적합하도록 이 파일을 편집할 수 있습니다. 설치된 Flex SDK의 frameworks 디렉토리에는 두 개의 전역 Flex 구성 파일이 있습니다. air-config.xml 파일은 amxmlc 컴파일러를 실행할 때 사용됩니다. 이 파일은 AIR 라이브러리를 포함하여 AIR에 대한 컴파일러를 구성합니다. flex-config.xml 파일은 mxmlc를 실행할 때 사용됩니다.

기본 구성 값은 Flex 및 AIR의 작동 방식을 확인할 때 적합하지만 전체 프로젝트를 시작할 때는 가능한 옵션들을 보다 세밀하게 검사해야 합니다. 특정 프로젝트에 대해 전역 값보다 우선 적용되는 로컬 구성 파일의 컴파일러 옵션에 프로젝트별 값을 제공할 수 있습니다.

참고: 컴파일 옵션은 특히 AIR 응용 프로그램에 대해 사용되지 않지만 AIR 응용 프로그램을 컴파일할 때는 AIR 라이브러리를 참조해야 합니다. 일반적으로 이러한 라이브러리는 프로젝트 레벨 구성, Ant와 같은 빌드 도구에 대한 파일 또는 명령줄에서 직접 참조됩니다.

AIR용 MXML 및 ActionScript 소스 파일 컴파일

다음과 같이 명령줄 MXML 컴파일러(amxmlc)를 사용하여 AIR 응용 프로그램의 Adobe® ActionScript® 3.0 및 MXML 에셋을 컴파일할 수 있습니다. HTML 기반 응용 프로그램은 컴파일할 필요가 없습니다. Flash Professional에서 SWF를 컴파일하려면 동영상을 SWF 파일로 제작하기만 하면 됩니다.

amxmlc를 사용하는 기본 명령줄 패턴은 다음과 같습니다.

```
amxmlc [compiler options] -- MyAIRApp.mxml
```

여기서 **[compiler options]**은 AIR 응용 프로그램을 컴파일하는 데 사용하는 명령줄 옵션을 지정합니다.

amxmlc 명령은 표준 Flex mxmlc 컴파일러와 추가 매개변수 +configname=air를 호출합니다. 이 매개변수는 컴파일러에게 flex-config.xml 파일 대신 air-config.xml 파일을 사용하도록 지시합니다. amxmlc를 사용하는 것은 mxmlc를 사용하는 것과 동일합니다.

컴파일러는 일반적으로 AIR 응용 프로그램을 컴파일하는 데 필요한 AIR 및 Flex 라이브러리를 지정하는 air-config.xml 구성 파일을 로드합니다. 또한 전역 구성에 대한 추가 옵션을 재정의하거나 추가하는 데 프로젝트 레벨의 로컬 구성 파일을 사용할 수도 있습니다. 일반적으로 로컬 구성 파일을 만드는 가장 쉬운 방법은 전역 버전의 사본을 편집하는 것입니다. -load-config 옵션을 사용하여 로컬 파일을 로드할 수 있습니다.

-load-config=project-config.xml 전역 옵션을 재정의합니다.

-load-config+=project-config.xml -library-path 옵션과 같이 값 이외의 것을 사용하는 전역 옵션에 추가 값을 추가합니다. 단일 값만을 사용하는 전역 옵션이 재정의됩니다.

로컬 구성 파일에 대한 특별한 이름 지정 규칙을 사용할 경우 amxmlc 컴파일러는 로컬 파일을 자동으로 로드합니다. 예를 들어 기본 MXML 파일이 RunningMan.mxml인 경우 로컬 구성 파일의 이름을 RunningMan-config.xml로 지정합니다. 이제 응용 프로그램을 컴파일하려면 다음을 입력하기만 하면 됩니다.

```
amxmlc RunningMan.mxml
```

파일 이름이 컴파일된 MXML 파일 이름과 일치하므로 RunningMan-config.xml이 자동으로 로드됩니다.

amxmlc 예제

다음 예제에서는 amxmlc 컴파일러의 사용을 보여 줍니다. 여기서는 응용 프로그램의 ActionScript 및 MXML 에셋만 컴파일됩니다.

AIR MXML 파일 컴파일:

```
amxmlc myApp.mxml
```

컴파일 및 출력 이름 설정:

```
amxmlc -output anApp.swf -- myApp.mxml
```

AIR ActionScript 파일 컴파일:

```
amxmlc myApp.as
```

컴파일러 구성 파일 지정:

```
amxmlc -load-config config.xml -- myApp.mxml
```

다른 구성 파일에서 추가 옵션 추가:

```
amxmlc -load-config+=moreConfig.xml -- myApp.mxml
```

명령줄에서 이미 구성 파일에 있는 라이브러리에 라이브러리 추가:

```
amxmlc -library-path+=/libs/libOne.swc,/libs/libTwo.swc  -- myApp.mxml
```

구성 파일을 사용하지 않고 AIR MXML 파일 컴파일(Win):

```
mxmlc -library-path [AIR SDK]/frameworks/libs/air/airframework.swc, ^
[AIR SDK]/frameworks/libs/air/airframework.swc, ^
-library-path [Flex SDK]/frameworks/libs/framework.swc ^
-- myApp.mxml
```

구성 파일을 사용하지 않고 AIR MXML 파일 컴파일(Mac OS X 또는 Linux):

```
mxmlc -library-path [AIR SDK]/frameworks/libs/air/airframework.swc, \
[AIR SDK]/frameworks/libs/air/airframework.swc, \
-library-path [Flex 3 SDK]/frameworks/libs/framework.swc \
-- myApp.mxml
```

런타임 공유 라이브러리를 사용하기 위한 AIR MXML 파일 컴파일:

```
amxmlc -external-library-path+=../lib/myLib.swc -runtime-shared-libraries=myrsl.swf -- myApp.mxml
```

ANE를 사용하기 위한 AIR MXML 파일 컴파일(ANE에 대해 반드시 -external-library-path 사용):

```
amxmlc -external-library-path+=../lib/myANE.ane -output=myAneApp.swf -- myAneApp.mxml
```

mxmlc.jar을 포함하도록 클래스 경로를 설정하여 Java에서 컴파일:

```
java flex2.tools.Compiler +flexlib [Flex SDK 3]/frameworks +configname=air [additional compiler options] -
- myApp.mxml
```

flexlib 옵션은 컴파일러에서 flex_config.xml 파일의 위치를 찾을 수 있도록 활성화시킴으로써 Flex SDK 프레임워크 디렉토리의 위치를 식별합니다.

클래스 경로를 설정하지 않고 Java에서 컴파일:

```
java -jar [Flex SDK 2]/lib/mxmlc.jar +flexlib [Flex SDK 3]/frameworks +configname=air [additional compiler
options] -- myApp.mxml
```

Apache Ant를 사용하는 컴파일러를 호출하려면(예제에서는 Java 작업을 사용하여 mxmlc.jar 실행)

```
<property name="SDK_HOME" value="C:/Flex46SDK"/>
<property name="MAIN_SOURCE_FILE" value="src/myApp.mxml"/>
<property name="DEBUG" value="true"/>
<target name="compile">
    <java jar="${MXMLC.JAR}" fork="true" failonerror="true">
        <arg value="-debug=${DEBUG}"/>
        <arg value="+flexlib=${SDK_HOME}/frameworks"/>
        <arg value="+configname=air"/>
        <arg value="-file-specs=${MAIN_SOURCE_FILE}"/>
    </java>
</target>
```

AIR 구성 요소 또는 코드 라이브러리 컴파일 (Flex)

구성 요소 컴파일러인 acompc를 사용하여 AIR 라이브러리 및 독립 구성 요소를 컴파일합니다. acompc 구성 요소 컴파일러는 amxmlc 컴파일러처럼 동작하지만 다음과 같은 예외가 있습니다.

- 라이브러리 또는 구성 요소에 포함시키려면 코드 베이스 내에 어떤 클래스인지를 지정해야 합니다.
- acompc는 로컬 구성 파일을 자동으로 검색하지 않습니다. 프로젝트 구성 파일을 사용하려면 –load-config 옵션을 사용해야 합니다.

acompc 명령은 표준 Flex compc 구성 요소 컴파일러를 호출하지만 해당 구성 옵션은 air-config.xml 파일에서 로드하며, flex-config.xml 파일에서 로드하지는 않습니다.

구성 요소 컴파일러 구성 파일

로컬 구성 파일을 사용하면 명령줄에서 소스 경로 및 클래스 이름을 입력하지 않아도 됩니다. 이는 잘못 입력될 가능성을 방지합니다. 로컬 구성 파일을 로드하려면 -load-config 옵션을 acompc 명령줄에 추가합니다.

다음 예제에서는 com.adobe.samples.particles 패키지의 두 개의 클래스인 ParticleManager 및 Particle을 사용하여 라이브러리를 구축하기 위한 구성을 보여 줍니다. 클래스 파일은 source/com/adobe/samples/particles 폴더에 들어 있습니다.

```
<flex-config>
    <compiler>
        <source-path>
            <path-element>source</path-element>
        </source-path>
    </compiler>
    <include-classes>
        <class>com.adobe.samples.particles.ParticleManager</class>
        <class>com.adobe.samples.particles.Particle</class>
    </include-classes>
</flex-config>
```

ParticleLib-config.xml이라는 구성 파일을 사용하여 라이브러리를 컴파일하려면 다음과 같이 입력합니다.

```
acompc -load-config ParticleLib-config.xml -output ParticleLib.swc
```

명령줄에서 동일한 명령을 실행하려면 다음과 같이 입력합니다.

```
acompc -source-path source -include-classes com.adobe.samples.particles.Particle
com.adobe.samples.particles.ParticleManager -output ParticleLib.swc
```

한 줄에 명령 전체를 입력하거나 해당 명령 셸의 줄 연결 문자를 사용합니다.

acompc 예제

이 예제에서는 myLib-config.xml이라는 구성 파일을 사용한다고 가정합니다.

AIR 구성 요소 또는 라이브러리 컴파일:

```
acompc -load-config myLib-config.xml -output lib/myLib.swc
```

런타임 공유 라이브러리 컴파일:

```
acompc -load-config myLib-config.xml -directory -output lib
```

이 명령을 실행하려면 lib 폴더가 존재해야 하며 비어 있어야 합니다.

11장: ADL(AIR Debug Launcher)

ADL(AIR Debug Launcher)을 사용하여 개발 중에 SWF 기반 응용 프로그램과 HTML 기반 응용 프로그램을 모두 실행합니다. ADL을 사용하면 먼저 응용 프로그램을 패키지 및 설치하지 않고도 실행할 수 있습니다. 기본적으로 ADL은 SDK에 포함된 런타임을 사용하므로 ADL을 사용하기 위해 런타임을 별도로 설치할 필요가 없습니다.

ADL은 trace 문 및 런타임 오류를 표준 출력으로 인쇄하지만 중단점 또는 기타 디버깅 기능은 지원하지 않습니다. 복잡한 디버깅 문제에는 Flash Debugger 또는 Flash Builder와 같은 통합 개발 환경을 사용할 수 있습니다.

참고: trace() 문이 콘솔에 표시되지 않는 경우, mm.cfg 파일에 ErrorReportingEnable 또는 TraceOutputFileEnable을 지정하지 않았는지 확인하십시오. 이 파일의 플랫폼별 위치에 대한 자세한 내용은 mm.cfg 파일 편집을 참조하십시오.

AIR는 디버깅을 직접 지원하므로 Adobe® Flash® Player에서와 같이 런타임의 디버그 버전이 필요하지 않습니다. 명령줄 디버깅을 수행하려면 Flash Debugger 및 ADL(AIR Debug Launcher)을 사용합니다.

Flash Debugger는 Flex SDK 디렉토리에 배포됩니다. Windows의 fdb.exe와 같은 기본 버전은 bin 하위 디렉토리에 있습니다. Java 버전은 lib 하위 디렉토리에 있습니다. AIR Debug Launcher인 adl.exe는 Flex SDK 설치의 bin 디렉토리에 있습니다. 별개의 Java 버전은 없습니다.

참고: fdb는 Flash Player로 시작하려고 시도하기 때문에 fdb로 직접 AIR 응용 프로그램을 시작할 수 없습니다. 대신 AIR 응용 프로그램이 실행 중인 fdb 세션에 연결하도록 하십시오.

ADL 사용

ADL로 응용 프로그램을 실행하려면 다음과 같은 패턴을 사용하십시오.

```
adl application.xml
```

여기서 **application.xml**은 응용 프로그램의 응용 프로그램 설명자 파일입니다.

ADL의 전체 구문은 다음과 같습니다.

```
adl [-runtime runtime-directory]
    [-pubid publisher-id]
    [-nodebug]
    [-atlogin]
    [-profile profileName]
    [-screensize value]
    [-extdir extension-directory]
    application.xml
    [root-directory]
    [-- arguments]
```

대괄호 [] 안의 항목은 선택 사항입니다.

-runtime runtime-directory 사용할 런타임이 포함된 디렉토리를 지정합니다. 지정하지 않을 경우 ADL 프로그램과 같은 SDK의 런타임 디렉토리가 사용됩니다. SDK 폴더에서 ADL을 다른 위치로 이동할 경우 런타임 디렉토리를 지정해야 합니다. Windows 및 Linux에서는 Adobe AIR 디렉토리가 포함된 디렉토리를 지정하십시오. Mac OS X에서는 Adobe AIR.framework가 포함된 디렉토리를 지정하십시오.

-pubid publisher-id 이 실행에 대한 AIR 응용 프로그램의 제작자 ID로 지정된 값을 지정합니다. 임시 제작자 ID를 지정함으로써 로컬 연결을 통한 통신과 같이 제작자 ID를 사용하여 응용 프로그램을 고유하게 식별하는 AIR 응용 프로그램의 기능을 테스트할 수 있습니다. AIR 1.5.3부터는 응용 프로그램 설명자 파일에서 제작자 ID를 지정할 수도 있습니다(이 매개 변수를 사용해서는 안 됨).

참고: AIR 1.5.3부터는 제작자 ID가 더 이상 자동으로 계산되어 AIR 응용 프로그램에 할당되지 않습니다. 기존 AIR 응용 프로그램에 대한 업데이트를 만들 때 제작자 ID를 지정할 수 있지만, 새 응용 프로그램에서는 제작자 ID를 지정할 필요가 없으며 지정해서도 안 됩니다.

-nodebug 디버깅 지원 기능을 해제합니다. 이를 사용할 경우 응용 프로그램 프로세스는 Flash 디버거에 연결되지 못하고 처리되지 않은 예외에 대한 대화 상자가 표시되지 않습니다. 그러나 trace 문은 계속 콘솔 윈도우에 인쇄됩니다. 디버깅 기능을 해제하면 응용 프로그램을 좀 더 빠르게 실행할 수 있으며 설치된 응용 프로그램의 실행 모드를 보다 근접하게 에뮬레이션할 수 있습니다.

-atlogin 로그인 시 응용 프로그램 실행을 시뮬레이션합니다. 이 플래그를 통해 사용자가 로그인하면 응용 프로그램이 실행되도록 설정된 경우에만 실행되는 응용 프로그램 논리를 테스트할 수 있습니다. -atlogin이 사용되는 경우 응용 프로그램에 전달되는 InvokeEvent 객체의 reason 속성은 standard가 아닌 login입니다(응용 프로그램이 아직 실행되지 않고 있는 경우).

-profile profileName ADL은 지정된 프로파일을 사용하여 응용 프로그램을 디버깅합니다. profileName은 다음 값 중 하나가 될 수 있습니다.

- desktop
- extendedDesktop
- mobileDevice

응용 프로그램 설명자에 supportedProfiles 요소가 있으면 -profile을 사용하여 지정하는 프로파일이 지원되는 목록에 포함되어 있어야 합니다. -profile 플래그를 사용하지 않으면 응용 프로그램 설명자에 있는 첫 번째 프로파일이 활성 프로파일로 사용됩니다. 응용 프로그램 설명자에 supportedProfiles 요소가 포함되어 있지 않을 때 -profile 플래그를 사용하지 않으면 **desktop** 프로파일이 사용됩니다.

자세한 내용은 216페이지의 “supportedProfiles” 및 222페이지의 “장치 프로파일”을 참조하십시오.

-screensize 값 데스크톱에서 mobileDevice 프로파일에 있는 응용 프로그램을 실행할 때 사용할 시뮬레이트한 스크린 크기입니다. 스크린 크기를 미리 정의된 스크린 유형 또는 세로 레이아웃의 보통 폭 및 높이에 대한 픽셀 크기로 지정한 후에 전체 화면 폭 및 높이를 지정하십시오. 유형을 기준으로 값을 지정하려면 다음과 같은 미리 정의된 스크린 유형 중 하나를 사용하십시오.

스크린 유형	보통 폭 x 높이	전체 화면 폭 x 높이
480	720 x 480	720 x 480
720	1280 x 720	1280 x 720
1080	1920 x 1080	1920 x 1080
Droid	480 x 816	480 x 854
FWQVGA	240 x 432	240 x 432
FWVGA	480 x 854	480 x 854
HVGA	320 x 480	320 x 480
iPad	768 x 1004	768 x 1024
iPadRetina	1536 x 2008	1536 x 2048
iPhone	320 x 460	320 x 480
iPhoneRetina	640 x 920	640 x 960
iPhone5Retina	640 x 1096	640 x 1136
iPhone6	750 x 1294	750 x 1334
iPhone6Plus	1242 x 2148	1242 x 2208
iPod	320 x 460	320 x 480

스크린 유형	보통 폭 x 높이	전체 화면 폭 x 높이
iPodRetina	640 x 920	640 x 960
iPod5Retina	640 x 1096	640 x 1136
NexusOne	480 x 762	480 x 800
QVGA	240 x 320	240 x 320
SamsungGalaxyS	480 x 762	480 x 800
SamsungGalaxyTab	600 x 986	600 x 1024
WQVGA	240 x 400	240 x 400
WVGA	480 x 800	480 x 800

스크린 픽셀 크기를 직접 지정하려면 다음 포맷을 사용하십시오.

```
widthXheight:fullscreenWidthXfullscreenHeight
```

항상 세로 레이아웃에 대한 픽셀 크기를 지정합니다. 이는 높이 값보다 작은 값을 폭에 지정해야 한다는 것을 의미합니다. 예를 들어 NexusOne 스크린은 다음을 사용하여 지정할 수 있습니다.

```
-screensize 480x762:480x800
```

-extdir extension-directory 런타임이 기본 확장을 검색해야 하는 디렉토리입니다. 이 디렉토리에는 응용 프로그램에 사용되는 각 기본 확장의 하위 디렉토리가 포함되어 있습니다. 이러한 각 하위 디렉토리에는 확장의 패키지화되지 않은 ANE 파일이 포함되어 있습니다. 예를 들면 다음과 같습니다.

```
C:\extensionDirs\
    extension1.ane\
        META-INF\
            ANE\
                Android-ARM\
                    library.swf
                    extension1.jar
                extension.xml
            signatures.xml
        catalog.xml
        library.swf
        mimetype
    extension2.ane\
        META-INF\
            ANE\
                Android-ARM\
                    library.swf
                    extension2.jar
                extension.xml
            signatures.xml
        catalog.xml
        library.swf
        mimetype
```

-extdir 매개 변수를 사용할 경우 다음 사항에 유의하십시오.

- ADL 명령을 사용하려면 지정된 각 디렉토리에 .ane 파일 이름 확장자가 있어야 합니다. 그러나 파일 이름에서 ".ane" 접미어 앞 부분은 유효한 아무 이름이나 지정할 수 있으며, 응용 프로그램 설명자 파일의 extensionID 요소 값과 일치할 필요가 없습니다.
- -extdir 매개 변수는 여러 번 지정할 수 있습니다.

- -extdir 매개 변수의 사용 방법은 ADT 도구와 ADL 도구 간에 서로 다릅니다. ADT에서 이 매개 변수는 ANE 파일이 포함된 디렉토리를 지정합니다.
- 또한 환경 변수 AIR_EXTENSION_PATH를 사용하여 확장 디렉토리를 지정할 수도 있습니다. 169페이지의 “ADT 환경 변수”를 참조하십시오.

application.xml 응용 프로그램 설명자 파일입니다. 183페이지의 “AIR 응용 프로그램 설명자 파일”을 참조하십시오. 응용 프로그램 설명자는 ADL의 유일한 필수 매개 변수이고, 대부분의 경우 이 매개 변수 외의 다른 매개 변수는 필요하지 않습니다.

root-directory 실행할 응용 프로그램의 루트 디렉토리를 지정합니다. 지정하지 않으면 응용 프로그램 설명자 파일이 포함된 디렉토리가 사용됩니다.

-- arguments "--" 다음에 나타나는 모든 문자열은 명령줄 인수로 응용 프로그램에 전달됩니다.

참고: 이미 실행 중인 AIR 응용 프로그램을 시작할 때 해당 응용 프로그램의 새 인스턴스가 시작되지 않습니다. 그 대신 invoke 이벤트가 실행 중인 인스턴스로 전달됩니다.

ADL 예제

현재 디렉토리에서 응용 프로그램 실행:

```
adl myApp-app.xml
```

현재 디렉토리의 하위 디렉토리에서 응용 프로그램 실행:

```
adl source/myApp-app.xml release
```

응용 프로그램을 실행하고 두 개의 명령줄 인수인 "tick"과 "tock"을 전달:

```
adl myApp-app.xml -- tick tock
```

특정 런타임을 사용하여 응용 프로그램 실행:

```
adl -runtime /AIRSDK/runtime myApp-app.xml
```

디버깅 지원 없이 응용 프로그램 실행:

```
adl -nodebug myApp-app.xml
```

휴대 장치 프로파일에서 응용 프로그램을 실행하고 Nexus One 스크린 크기를 시뮬레이트합니다.

```
adl -profile mobileDevice -screensize NexusOne myMobileApp-app.xml
```

Apache Ant를 사용하는 응용 프로그램을 실행하여 이 응용 프로그램을 실행합니다(예제에 표시된 경로는 Windows에 해당함).

```
<property name="SDK_HOME" value="C:/AIRSDK"/>
<property name="ADL" value="${SDK_HOME}/bin/adl.exe"/>
<property name="APP_ROOT" value="c:/dev/MyApp/bin-debug"/>
<property name="APP_DESCRIPTOR" value="${APP_ROOT}/myApp-app.xml"/>

<target name="test">
    <exec executable="${ADL}">
        <arg value="${APP_DESCRIPTOR}"/>
        <arg value="${APP_ROOT}"/>
    </exec>
</target>
```

ADL 종료 및 오류 코드

다음 표에서는 ADL이 인쇄하는 종료 코드에 대해 설명합니다.

종료 코드	설명
0	성공적으로 시작했습니다. AIR 응용 프로그램 종료 후 ADL이 종료됩니다.
1	이미 실행 중인 AIR 응용 프로그램을 성공적으로 호출하였습니다. ADL이 즉시 종료됩니다.
2	사용 오류입니다. ADL에 잘못된 인수가 제공되었습니다.
3	런타임을 찾을 수 없습니다.
4	런타임을 시작할 수 없습니다. 이는 종종 응용 프로그램에 지정된 버전이 런타임의 버전과 일치하지 않기 때문에 발생합니다.
5	원인을 알 수 없는 오류가 발생했습니다.
6	응용 프로그램 설명자 파일을 찾을 수 없습니다.
7	응용 프로그램 설명자의 내용이 유효하지 않습니다. 이 오류는 일반적으로 xml 형식이 올바르지 않음을 나타냅니다.
8	응용 프로그램 설명자 파일의 <content> 요소에 지정된 기본 응용 프로그램 내용 파일을 찾을 수 없습니다.
9	기본 응용 프로그램 내용 파일이 올바른 SWF 또는 HTML 파일이 아닙니다.
10	응용 프로그램에서 -profile 옵션으로 지정된 프로필을 지원하지 않습니다.
11	-screensize 인수는 현재 프로파일에서 지원되지 않습니다.

12장: ADT(AIR Developer Tool)

ADT(AIR Developer Tool)는 AIR 응용 프로그램을 개발하기 위한 다목적 명령줄 도구입니다. ADT를 사용하여 다음 작업을 수행할 수 있습니다.

- AIR 응용 프로그램을 .air 설치 파일로 패키지화
- AIR 응용 프로그램을 기본 설치 프로그램으로 패키지화(예: Windows의 .exe, iOS의 .ipa, Android의 .apk 설치 프로그램 파일)
- 기본 확장을 ANE(AIR Native Extension) 파일로 패키지화
- 디지털 인증서를 사용하여 AIR 응용 프로그램 서명
- 응용 프로그램 업데이트에 사용되는 디지털 서명 변경(마이그레이션)
- 컴퓨터에 연결된 장치 확인
- 자체 서명된 디지털 코드 서명 인증서 만들기
- 휴대 장치에서 응용 프로그램을 원격으로 설치, 실행 및 제거
- 휴대 장치에서 AIR 런타임을 원격으로 설치 및 제거

ADT는 AIR SDK에 포함된 Java 프로그램입니다. 사용하려면 Java 1.5 이상이 필요합니다. SDK에는 ADT를 호출하기 위한 스크립트 파일이 포함되어 있습니다. 이 스크립트를 사용하려면 Java 프로그램의 위치가 Path 환경 변수에 포함되어 있어야 합니다. AIR SDK bin 디렉토리도 Path 환경 변수에 포함되어 있는 경우에는 명령줄에 adt를 적절한 인수와 함께 입력하여 ADT를 호출할 수 있습니다. Path 환경 변수를 설정하는 방법을 모르면 운영 체제 설명서를 참조하십시오. 원활한 지원을 위해 대부분의 컴퓨터 시스템에 대한 경로를 설정하는 절차가 277페이지의 “path 환경 변수”에 설명되어 있습니다.

ADT를 사용하려면 최소한 2GB의 컴퓨터 메모리가 필요합니다. 이보다 메모리가 적으면 iOS용 응용 프로그램을 패키지화하는 경우 등에 ADT에서 메모리가 부족한 현상이 발생할 수 있습니다.

Java와 AIR SDK bin 디렉토리가 모두 Path 변수에 포함되어 있으면 다음과 같은 기본 구문을 사용하여 ADT를 실행할 수 있습니다.

```
adt -command options
```

참고: Adobe Flash Builder 및 Adobe Flash Professional을 비롯한 대부분의 통합 개발 환경에서 AIR 응용 프로그램을 패키지화하고 서명할 수 있습니다. 이와 같은 개발 환경을 사용하고 있다면 이러한 공통적인 작업에 ADT를 사용할 필요가 없습니다. 하지만 통합 개발 환경에서 지원하지 않는 기능의 경우에는 여전히 ADT를 명령줄 도구로 사용해야 합니다. 또한 자동화된 구성 프로세스의 일환으로 ADT를 명령줄 도구로 사용할 수 있습니다.

ADT 명령

ADT에 전달된 첫 번째 인수는 다음 명령 중 하나를 지정합니다.

- -package - AIR 응용 프로그램 또는 ANE(AIR Native Extension)를 패키지화합니다.
- -prepare - AIR 응용 프로그램을 중간 파일(AIRI)로 패키지화하지만 서명하지는 않습니다. AIRI 파일은 설치될 수 없습니다.
- -sign - -prepare 명령을 사용하여 만들어진 AIRI 패키지에 서명합니다. -prepare 및 -sign 명령을 사용하면 패키지화 및 서명을 서로 다른 시간에 수행할 수 있습니다. -sign 명령을 사용하여 ANE 패키지를 서명하거나 재서명할 수도 있습니다.
- -migrate - 서명된 AIR 패키지에 마이그레이션 서명을 적용함으로써 새로운 또는 갱신된 코드 서명 인증서를 사용할 수 있습니다.

- -certificate - 자체 서명된 디지털 코드 서명 인증서를 만듭니다.
- -checkstore - 키 저장소에 있는 디지털 인증서에 액세스할 수 있는지를 확인합니다.
- -installApp - 장치 또는 장치 에뮬레이터에 AIR 응용 프로그램을 설치합니다.
- -launchApp - 장치 또는 장치 에뮬레이터에서 AIR 응용 프로그램을 실행합니다.
- -appVersion - 장치 또는 장치 에뮬레이터에 현재 설치된 AIR 응용 프로그램의 버전을 보고합니다.
- -uninstallApp - 장치 또는 장치 에뮬레이터에서 AIR 응용 프로그램을 제거합니다.
- -installRuntime - 장치 또는 장치 에뮬레이터에 AIR 런타임을 설치합니다.
- -runtimeVersion - 장치 또는 장치 에뮬레이터에 현재 설치된 AIR 런타임의 버전을 보고합니다.
- -uninstallRuntime - 장치 또는 장치 에뮬레이터에서 현재 설치된 AIR 런타임을 제거합니다.
- -version - ADT 버전 번호를 보고합니다.
- -devices - 연결된 휴대 장치 또는 에뮬레이터의 장치 정보를 보고합니다.
- -help - 명령 및 옵션의 목록을 표시합니다.

많은 ADT 명령이 관련 옵션 플래그 및 매개 변수 집합을 공유합니다. 이러한 옵션 집합은 따로 자세히 설명되어 있습니다.

- 161페이지의 “ADT 코드 서명 옵션”
- 163페이지의 “파일 및 경로 옵션”
- 164페이지의 “디버거 연결 옵션”
- 165페이지의 “기본 확장 옵션”

ADT package 명령

-package 명령은 기본 응용 프로그램 디렉토리에서 실행해야 합니다. 이 명령은 다음 구문을 사용합니다.

구성 요소 응용 프로그램 파일로부터 AIR 패키지 만들기:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    -sampler
    -hideAneLibSymbols
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    FILE_OPTIONS
```

구성 요소 응용 프로그램 파일로부터 기본 패키지 만들기:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    DEBUGGER_CONNECTION_OPTIONS
    -airDownloadURL URL
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    -platformsdk path
    FILE_OPTIONS
```

구성 요소 응용 프로그램 파일에서 기본 확장이 포함된 기본 패키지를 만듭니다.

```
adt -package
    AIR_SIGNING_OPTIONS
    -migrate MIGRATION_SIGNING_OPTIONS
    -target packageType
    DEBUGGER_CONNECTION_OPTIONS
    -airDownloadURL URL
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    -platformsdk path
    FILE_OPTIONS
```

AIR 또는 AIRI 파일로부터 기본 패키지 만들기:

```
adt -package
    -target packageType
    NATIVE_SIGNING_OPTIONS
    output
    input_package
```

구성 요소 기본 확장 파일에서 기본 확장 패키지를 만듭니다.

```
adt -package
    AIR_SIGNING_OPTIONS
    -target ane
    output
    ANE_OPTIONS
```

참고: ANE 파일에 서명할 필요가 없으므로 이 예제에서 AIR_SIGNING_OPTIONS 매개 변수는 선택 사항입니다.

AIR_SIGNING_OPTIONS AIR 서명 옵션은 AIR 설치 파일에 서명하는 데 사용되는 인증서를 식별합니다. 서명 옵션은 161 페이지의 “ADT 코드 서명 옵션”에서 자세히 설명합니다.

-migrate 이 플래그는 응용 프로그램을 AIR_SIGNING_OPTIONS 매개 변수에 지정된 인증서뿐 아니라 마이그레이션 인증서도 사용하여 서명하도록 지정합니다. 이 플래그는 데스크톱 응용 프로그램을 기본 설치 프로그램으로 패키지화하고 해당 응용 프로그램이 기본 확장을 사용하는 경우에만 유효합니다. 그 외 경우는 오류가 발생합니다. 마이그레이션 인증서의 서명 옵션은 **MIGRATION_SIGNING_OPTIONS** 매개 변수로 지정됩니다. 서명 옵션은 161페이지의 “ADT 코드 서명 옵션”에서 자세히 설명합니다. -migrate 플래그를 사용하면 기본 확장을 사용하는 데스크톱 기본 설치 프로그램의 업데이트를 만들 수 있고 원래 인증서가 만료되는 등의 경우에 응용 프로그램의 코드 서명 인증서를 변경할 수 있습니다. 자세한 내용은 179페이지의 “업데이트된 버전의 AIR 응용 프로그램에 서명”을 참조하십시오.

-package 명령의 -migrate 플래그는 AIR 3.6 이상에서 사용할 수 있습니다.

-target 만들 패키지의 유형입니다. 지원되는 패키지 유형은 다음과 같습니다.

- air - AIR 패키지입니다. “air”는 기본값이며, AIR 또는 AIRI 파일을 만들 때는 -target 플래그를 지정하지 않아도 됩니다.
- airn - 확장 TV 프로파일에 있는 장치에 대한 기본 응용 프로그램 패키지입니다.
- ane - AIR 기본 확장 패키지입니다.
- Android 패키지의 대상은 다음과 같습니다.
  - apk - Android 패키지입니다. 이 target으로 생성되는 패키지는 에뮬레이터가 아니라 Android 장치에만 설치될 수 있습니다.
  - apk-captive-runtime - 응용 프로그램과 전용 버전의 AIR 런타임이 모두 포함된 Android 패키지입니다. 이 target으로 생성되는 패키지는 에뮬레이터가 아니라 Android 장치에만 설치될 수 있습니다.
  - apk-debug - 추가 디버깅 정보가 있는 Android 패키지입니다. 응용 프로그램에 있는 SWF 파일은 디버깅 지원과도 함께 컴파일되어야 합니다.
  - apk-emulator - 디버깅 지원이 없는 에뮬레이터에서 사용할 Android 패키지입니다. apk-debug target을 사용하여 에뮬레이터와 장치 모두에서 디버깅을 허용하십시오.

  - apk-profile - 응용 프로그램 성능 및 메모리 프로파일링을 지원하는 Android 패키지입니다.
- iOS 패키지의 대상은 다음과 같습니다.
  - ipa-ad-hoc - 애드혹 배포를 위한 iOS 패키지입니다.
  - ipa-app-store - Apple App 저장소 배포를 위한 iOS 패키지입니다.
  - ipa-debug - 추가 디버깅 정보가 있는 iOS 패키지입니다. 응용 프로그램에 있는 SWF 파일은 디버깅 지원과도 함께 컴파일되어야 합니다.
  - ipa-test - 최적화 또는 디버깅 정보 없이 컴파일된 iOS 패키지입니다.
  - ipa-debug-interpreter - 기능 면에서 디버그 패키지와 동일하지만 컴파일 속도가 더 빠릅니다. 그러나 ActionScript 바이트코드가 해석되기는 하지만 컴퓨터 코드로 변환되지는 않습니다. 따라서 인터프리터 패키지에서는 코드 실행 속도가 느립니다.
  - ipa-debug-interpreter-simulator - 기능 면에서 ipa-debug-interpreter와 동일하지만 iOS 시뮬레이터용으로 패키지화되었습니다. Macintosh 전용입니다. 이 옵션을 사용하는 경우 -platformsdk 옵션도 포함하여 iOS 시뮬레이터 SDK의 경로를 지정해야 합니다.
  - ipa-test-interpreter - 기능 면에서 테스트 패키지와 동일하지만 컴파일 속도가 더 빠릅니다. 그러나 ActionScript 바이트코드가 해석되기는 하지만 컴퓨터 코드로 변환되지는 않습니다. 따라서 인터프리터 패키지에서는 코드 실행 속도가 느립니다.
  - ipa-test-interpreter-simulator - 기능 면에서 ipa-test-interpreter와 동일하지만 iOS 시뮬레이터용으로 패키지화되었습니다. Macintosh 전용입니다. 이 옵션을 사용하는 경우 -platformsdk 옵션도 포함하여 iOS 시뮬레이터 SDK의 경로를 지정해야 합니다.
- native - 기본 데스크톱 설치 프로그램입니다. 생성되는 파일 유형은 명령이 실행되는 운영 체제의 기본 설치 포맷입니다.
  - EXE - Windows
  - DMG - Mac
  - DEB - Ubuntu Linux (AIR 2.6 이하)
  - RPM - Fedora 또는 OpenSuse Linux(AIR 2.6 이하)

  자세한 내용은 51페이지의 “데스크톱 기본 설치 프로그램 패키지화”를 참조하십시오.

-sampler(iOS 전용, AIR 3.4 이상) iOS 응용 프로그램에서 원격 측정 기반 ActionScript 샘플러를 사용할 수 있도록 합니다. 이 플래그를 사용하면 Adobe Scout를 통해 응용 프로그램을 프로파일링할 수 있습니다. Scout를 활용해서 모든 Flash 플랫폼 내용을 프로파일링할 수도 있지만 자세한 원격 측정을 사용하면 ActionScript 함수 시간, DisplayList, Stage3D 렌더링 등을 비롯한 여러 항목을 면밀히 파악할 수 있습니다. 이 플래그는 성능에 약간의 영향을 줄 수 있으므로 업무용 응용 프로그램에는 사용하지 않습니다.

-hideAneLibSymbols(iOS 전용, AIR 3.4 이상) 응용 프로그램 개발자는 여러 소스로부터 여러 기본 확장을 사용할 수 있으며, ANE 간에 공통된 심볼 이름을 공유할 경우 ADT에서 “객체 파일 내 중복된 심볼” 오류를 생성합니다. 경우에 따라 이 오류는 런타임에 충돌로 나타날 수도 있습니다. hideAneLibSymbols 옵션을 사용하면 ANE 라이브러리의 심볼을 해당 라이브러리의 소스에서만 볼 수 있도록 지정(yes)하거나 전역에 보이도록 지정(no)할 수 있습니다.

- **yes** - ANE 심볼을 숨기며, 이 경우 의도하지 않은 모든 심볼 충돌 문제가 해결됩니다.
- **no** - 기본값이며, ANE 심볼을 숨기지 않습니다. 사전 AIR 3.4 비헤이비어입니다.

-embedBitcode(iOS만 해당, AIR 25 이상) 응용 프로그램 개발자는 embedBitcode 옵션을 사용하여 yes 또는 no를 지정하는 방식으로 자신의 iOS 응용 프로그램에 비트코드를 포함할 것인지 여부를 지정할 수 있습니다. 지정하지 않는 경우 이 스위치의 기본값은 no입니다.

DEBUGGER_CONNECTION_OPTIONS 디버거 연결 옵션은 디버그 패키지가 다른 컴퓨터에서 실행 중인 원격 디버거에 연결을 시도해야 하는지 아니면 원격 디버거로부터 연결을 수신 대기해야 하는지를 지정합니다. 이 옵션 집합은 모바일 디버그 패키지(apk-debug 및 ipa-debug를 대상으로 함)에 대해서만 지원됩니다. 이들 옵션은 164페이지의 "디버거 연결 옵션"에 설명되어 있습니다.

-airDownloadURL Android 장치에 AIR 런타임을 다운로드하고 설치하기 위한 대체 URL을 지정합니다. 지정하지 않으면 런타임이 아직 설치되지 않은 경우 AIR 응용 프로그램이 사용자를 Android Market에 있는 AIR 런타임으로 리디렉션합니다.

응용 프로그램이 대체 마켓플레이스(Google에서 관리하는 Android Market이 아닌 마켓플레이스)를 통해 배포되는 경우에는 해당 마켓에서 AIR 런타임을 다운로드하기 위한 URL을 지정해야 할 수도 있습니다. 일부 대체 마켓에서는 응용 프로그램이 해당 마켓 외부로부터의 다운로드를 요구하는 것을 허용하지 않습니다. 이 옵션은 Android 패키지에 대해서만 지원됩니다.

NATIVE_SIGNING_OPTIONS 기본 서명 옵션은 기본 패키지 파일에 서명하는 데 사용되는 인증서를 식별합니다. 이러한 서명 옵션은 AIR 런타임이 아니라 기본 운영 체제에서 사용하는 서명을 적용하는 데 사용됩니다. 이 옵션은 그렇지 않을 경우 AIR_SIGNING_OPTIONS와 동일하며, 161페이지의 "ADT 코드 서명 옵션"에 자세히 설명되어 있습니다.

기본 서명은 Windows와 Android에서 지원됩니다. Windows에서는 AIR 서명 옵션과 기본 서명 옵션을 모두 지정해야 합니다. Android에서는 기본 서명 옵션만 지정할 수 있습니다.

종종 같은 코드 서명 인증서를 사용하여 AIR 및 기본 서명을 모두 적용할 수 있습니다. 하지만 언제나 가능한 것은 아닙니다. 예를 들어 Android Market에 제출된 Google의 응용 프로그램 정책은 모든 응용 프로그램이 적어도 2033년까지 유효한 인증서를 사용하여 서명될 것을 요구합니다. 즉, AIR 서명을 적용할 때 권장되는 유명 인증 기관에서 발행한 인증서는 Android 응용 프로그램에 서명하는 데 사용해서는 안 됩니다. 이처럼 유효 기간이 긴 코드 서명 인증서를 발행하는 인증 기관은 없기 때문입니다.

output 만들 패키지 파일의 이름입니다. 파일 확장자를 지정하는 것은 선택 사항입니다. 지정하지 않으면 -target 값과 현재 운영 체제에 적합한 확장자가 추가됩니다.

app_descriptor 응용 프로그램 설명자 파일에 대한 경로입니다. 경로는 현대 디렉토리에 대한 상대 경로로 지정하거나 절대 경로로 지정할 수 있습니다. 응용 프로그램 설명자 파일은 AIR 파일에서 **application.xml**로 이름이 변경됩니다.

-platformsdk 대상 장치의 플랫폼 SDK에 대한 경로입니다.

- Android - AIR 2.6 이상의 SDK에는 관련 ADT 명령을 구현하는 데 필요한 Android SDK의 도구가 포함되어 있습니다. 다른 버전의 Android SDK를 사용하려는 경우에만 이 값을 설정하십시오. 또한 AIR_ANDROID_SDK_HOME 환경 변수가 이미 설정된 경우에는 명령줄에 플랫폼 SDK 경로를 입력할 필요가 없습니다. 둘 다 설정되어 있으면 명령줄에 입력된 경로가 사용됩니다.
- iOS - AIR SDK가 전용 iOS SDK와 함께 제공됩니다. -platformsdk 옵션을 사용하면 외부 SDK를 사용하여 응용 프로그램을 패키지화할 수 있으므로 전용 iOS SDK만 사용해야 하는 문제가 없습니다. 예를 들어 최신 iOS SDK를 사용하여 확장을 빌드한 경우 응용 프로그램을 패키지화할 때 해당 SDK를 지정할 수 있습니다. 또한 iOS 시뮬레이터와 함께 ADT를 사용하는 경우에는 항상 -platformsdk 옵션을 포함하여 iOS 시뮬레이터 SDK의 경로를 지정해야 합니다.

-arch 응용 프로그램 개발자는 이 인수를 사용하여 x86 플랫폼용 APK를 만들 수 있으며, 여기에는 다음 값이 사용됩니다.

- armv7 - Android armv7 플랫폼용 ADT 패키지 APK입니다.
- x86 - Android x86 플랫폼용 ADT 패키지 APK입니다.

지정된 값이 없는 경우에는 armv7이 기본값입니다.

FILE_OPTIONS 패키지에 포함할 응용 프로그램 파일을 식별합니다. 파일 옵션은 163페이지의 "파일 및 경로 옵션"에서 자세히 설명합니다. AIR 또는 AIRI 파일로부터 기본 패키지를 만들 때는 파일 옵션을 지정하지 마십시오.

input_airi AIRI 파일로부터 기본 패키지를 만들 때 지정합니다. AIR_SIGNING_OPTIONS는 target이 **air**인 경우(또는 target이 지정되지 않은 경우)에 필요합니다.

input_air AIR 파일로부터 기본 패키지를 만들 때 지정합니다. AIR_SIGNING_OPTIONS는 지정하지 마십시오.

ANE_OPTIONS 기본 확장 패키지를 만들기 위한 옵션과 파일을 식별합니다. 확장 패키지 옵션은 165페이지의 "기본 확장 옵션"에서 자세히 설명합니다.

ADT -package 명령 예제

SWF 기반 AIR 응용 프로그램에 대해 현재 디렉토리에 있는 특정 응용 프로그램 파일을 패키지:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf components.swc
```

HTML 기반 AIR 응용 프로그램에 대해 현재 디렉토리에 있는 특정 응용 프로그램 파일을 패키지:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html AIRAliases.js image.gif
```

현재 작업 디렉토리의 모든 파일 및 하위 디렉토리 패키지:

```
adt –package -storetype pkcs12 -keystore ../cert.p12 myApp.air myApp.xml .
```

참고: 키 저장소 파일에는 응용 프로그램 서명에 사용되는 개인 키가 들어 있습니다. AIR 패키지 안에 서명 인증서를 포함시키지 마십시오. ADT 명령에 와일드카드를 사용할 경우 키 저장소 파일을 다른 위치에 보관하여 패키지에 포함되지 않도록 하십시오. 이 예제에서 키 저장소 파일 cert.p12는 상위 디렉토리에 있습니다.

기본 파일 및 이미지 하위 디렉토리만 패키지:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf images
```

HTML 기반 응용 프로그램 및 HTML, 스크립트 및 이미지 하위 디렉토리의 모든 파일을 패키지:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml index.html AIRALiases.js html scripts
images
```

작업 디렉토리(release/bin)에 있는 application.xml 파일 및 기본 SWF 패키지:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml –C release/bin myApp.swf
```

빌드 파일 시스템의 두 곳 이상에서 에셋을 패키지화합니다. 이 예제에서 응용 프로그램 에셋은 패키징되기 전에 다음과 같은 폴더에 들어 있습니다.

```
/devRoot
    /myApp
        /release
            /bin
                myApp-app.xml
                myApp.swf or myApp.html
    /artwork
        /myApp
            /images
                image-1.png
                ...
                image-n.png
    /libraries
        /release
            /libs
                lib-1.swf
                lib-2.swf
                lib-a.js
                AIRAliases.js
```

/devRoot/myApp 디렉토리에서 다음과 같은 ADT 명령 실행:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp-app.xml
    –C release/bin myApp.swf (or myApp.html)
    –C ../artwork/myApp images
    –C ../libraries/release libs
```

다음 패키지 구조 생성:

```
/myAppRoot
    /META-INF
        /AIR
            application.xml
            hash
    myApp.swf or myApp.html
    mimetype
    /images
        image-1.png
        ...
        image-n.png
    /libs
        lib-1.swf
        lib-2.swf
        lib-a.js
        AIRAliases.js
```

단순 SWF 기반 응용 프로그램에 대해 Java 프로그램으로 ADT 실행(클래스 경로 설정 안 함):

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf
```

단순 HTML 기반 응용 프로그램에 대해 Java 프로그램으로 ADT 실행(클래스 경로 설정 안 함):

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html
AIRAliases.js
```

ADT.jar 패키지를 포함하도록 Java 클래스 경로를 설정하여 Java 프로그램으로 ADT 실행:

```
java -com.adobe.air.ADT –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf
```

Apache Ant에서 Java 작업으로 ADT를 실행합니다. 그러나 일반적으로 Ant 스크립트에서 직접 ADT 명령을 사용하는 것이 가장 좋습니다. 다음 예제에 표시된 경로는 Windows의 경로입니다.

```
<property name="SDK_HOME" value="C:/AIRSDK"/>
<property name="ADT.JAR" value="${SDK_HOME}/lib/adt.jar"/>

target name="package">
    <java jar="${ADT.JAR}" fork="true" failonerror="true">
        <arg value="-package"/>
        <arg value="-storetype"/>
        <arg value="pkcs12"/>
        <arg value="-keystore"/>
        <arg value="../../ExampleCert.p12"/>
        <arg value="myApp.air"/>
        <arg value="myApp-app.xml"/>
        <arg value="myApp.swf"/>
        <arg value="icons/*.png"/>
    </java>
</target>
```

참고: 일부 컴퓨터 시스템에서는 파일 시스템 경로에 있는 2바이트 문자가 잘못 해석될 수 있습니다. 그럴 경우에는 ADT를 실행하는 데 사용된 JRE이 UTF-8 문자 세트를 사용하도록 설정해 보십시오. 이 작업은 기본적으로 Mac 및 Linux에서 ADT를 실행하는 데 사용하는 스크립트에서 수행할 수 있습니다. Windows adt.bat 파일에서 또는 Java에서 직접 ADT를 실행할 때 Java 명령줄에 -Dfile.encoding=UTF-8 옵션을 지정하십시오.

ADT prepare 명령

-prepare 명령은 서명되지 않은 AIRI 패키지를 만듭니다. AIRI 패키지는 독립적으로 사용할 수 없습니다. -sign 명령을 사용하여 AIRI 파일을 서명된 AIR 패키지로 변환하거나, package 명령을 사용하여 AIRI 파일을 기본 패키지로 변환하십시오.

-prepare 명령은 다음 구문을 사용합니다.

```
adt -prepare output app_descriptor FILE_OPTIONS
```

output 생성된 AIRI 파일의 이름입니다.

app_descriptor 응용 프로그램 설명자 파일에 대한 경로입니다. 경로는 현대 디렉토리에 대한 상대 경로로 지정하거나 절대 경로로 지정할 수 있습니다. 응용 프로그램 설명자 파일은 AIR 파일에서 **application.xml**로 이름이 변경됩니다.

FILE_OPTIONS 패키지에 포함할 응용 프로그램 파일을 식별합니다. 파일 옵션은 163페이지의 "파일 및 경로 옵션"에서 자세히 설명합니다.

ADT sign 명령

-sign 명령은 AIRI 및 ANE 파일에 서명합니다.

-sign 명령은 다음 구문을 사용합니다.

```
adt -sign AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS AIR 서명 옵션은 패키지 파일에 서명하는 데 사용되는 인증서를 식별합니다. 서명 옵션은 161페이지의 "ADT 코드 서명 옵션"에서 자세히 설명합니다.

input 서명할 AIRI 또는 ANE 파일의 이름입니다.

output 만들 서명된 패키지의 이름입니다.

ANE 파일이 이미 서명되어 있는 경우 이전 서명이 삭제됩니다. AIR 파일을 재서명할 수는 없습니다. 응용 프로그램 업데이트에 대해 새 서명을 사용하려면 -migrate 명령을 사용하십시오.

ADT migrate 명령

-migrate 명령은 마이그레이션 서명을 AIR 파일에 적용합니다. 디지털 인증서를 갱신하거나 변경해서 이전 인증서로 서명한 응용 프로그램을 업데이트해야 하는 경우 마이그레이션 서명을 사용해야 합니다.

마이그레이션 서명을 사용한 AIR 응용 프로그램 패키지화에 관한 자세한 내용은 179페이지의 "업데이트된 버전의 AIR 응용 프로그램에 서명"을 참조하십시오.

참고: 마이그레이션 인증서는 인증서가 만료된 날짜로부터 365일 이내에 적용되어야 합니다. 이 유예 기간이 경과되면 더 이상 마이그레이션 서명을 사용하여 응용 프로그램 업데이트에 서명할 수 없습니다. 사용자는 먼저 마이그레이션 서명을 사용하여 서명된 응용 프로그램의 버전으로 업데이트한 후에 최신 업데이트를 설치하거나, 원래 응용 프로그램을 제거하고 새 AIR 패키지를 설치할 수 있습니다.

마이그레이션 서명을 사용하려면 먼저 새로운 또는 갱신된 인증서를 사용하여 AIR 응용 프로그램에 서명(-package 또는 -sign 명령 사용)한 다음 이전 인증서와 -migrate 명령을 사용하여 마이그레이션 서명을 적용하십시오.

-migrate 명령은 다음 구문을 사용합니다.

```
adt -migrate AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS 기존 버전의 AIR 응용 프로그램에 서명하는 데 사용된 원래 인증서를 식별하는 AIR 서명 옵션입니다. 서명 옵션은 161페이지의 "ADT 코드 서명 옵션"에서 자세히 설명합니다.

input 새로운 응용 프로그램 인증서를 사용하여 이미 서명한 AIR 파일입니다.

output 새 인증서와 이전 인증서 모두의 서명을 포함하고 있는 최종 패키지의 이름입니다.

입력 및 출력 AIR 파일에 사용되는 파일 이름은 서로 달라야 합니다.

참고: 기본 확장이 포함된 AIR 데스크톱 응용 프로그램은 .air 파일이 아닌 기본 설치 프로그램으로 패키지화되기 때문에 이 AIR 데스크톱 응용 프로그램에는 ADT migrate 명령을 사용할 수 없습니다. 기본 확장이 포함된 AIR 데스크톱 응용 프로그램의 인증서를 변경하려면 150페이지의 "ADT package 명령"을 -migrate 플래그와 함께 사용하여 응용 프로그램을 패키지화합니다.

ADT checkstore 명령

-checkstore 명령을 사용하면 키 저장소의 유효성을 검사할 수 있습니다. 이 명령은 다음 구문을 사용합니다.

```
adt -checkstore SIGNING_OPTIONS
```

SIGNING_OPTIONS 유효성을 검사할 키 저장소를 식별하는 서명 옵션입니다. 서명 옵션은 161페이지의 “ADT 코드 서명 옵션”에서 자세히 설명합니다.

ADT certificate 명령

-certificate 명령을 사용하면 자체 서명된 디지털 코드 서명 인증서를 만들 수 있습니다. 이 명령은 다음 구문을 사용합니다.

```
adt -certificate -cn name -ou orgUnit -o orgName -c country -validityPeriod years key-type output password
```

-cn 새 인증서의 공용 이름으로 지정된 문자열입니다.

-ou 인증서를 발급한 기관 구성 단위로 지정된 문자열입니다. 이는 선택 사항입니다.

-o 인증서를 발급한 기관으로 지정된 문자열입니다. 이는 선택 사항입니다.

-c 2자로 구성된 ISO-3166 국가 코드입니다. 코드가 유효하지 않을 경우 인증서가 생성되지 않습니다. 이는 선택 사항입니다.

-validityPeriod 인증서의 유효성이 유지되는 연 단위의 기간입니다. 지정하지 않으면 5년의 유효 기간이 할당됩니다. 이는 선택 사항입니다.

key_type 인증서에 사용할 키 유형으로, **2048-RSA**입니다.

output 생성될 인증서 파일의 경로 및 파일 이름입니다.

password 새 인증서에 액세스하는 데 사용할 암호입니다. 이 인증서로 AIR 파일에 서명할 때 암호가 필요합니다.

ADT installApp 명령

-installApp 명령은 장치 또는 에뮬레이터에 응용 프로그램을 설치합니다.

먼저 기존 응용 프로그램을 제거한 후에야 이 명령을 사용하여 다시 설치할 수 있습니다.

이 명령은 다음 구문을 사용합니다.

```
adt -installApp -platform platformName -platformsdk path-to-sdk -device deviceID -package fileName
```

-platform 장치 플랫폼의 이름입니다. **ios** 또는 **android**를 지정합니다.

-platformsdk 대상 장치의 플랫폼 SDK에 대한 경로입니다(선택 사항).

- Android - AIR 2.6 이상의 SDK에는 관련 ADT 명령을 구현하는 데 필요한 Android SDK의 도구가 포함되어 있습니다. 다른 버전의 Android SDK를 사용하려는 경우에만 이 값을 설정하십시오. 또한 AIR_ANDROID_SDK_HOME 환경 변수가 이미 설정된 경우에는 명령줄에 플랫폼 SDK 경로를 입력할 필요가 없습니다. 둘 다 설정되어 있으면 명령줄에 입력된 경로가 사용됩니다.
- iOS - AIR SDK가 전용 iOS SDK와 함께 제공됩니다. -platformsdk 옵션을 사용하면 외부 SDK를 사용하여 응용 프로그램을 패키지화할 수 있으므로 전용 iOS SDK만 사용해야 하는 문제가 없습니다. 예를 들어 최신 iOS SDK를 사용하여 확장을 빌드한 경우 응용 프로그램을 패키지화할 때 해당 SDK를 지정할 수 있습니다. 또한 iOS 시뮬레이터와 함께 ADT를 사용하는 경우에는 항상 -platformsdk 옵션을 포함하여 iOS 시뮬레이터 SDK의 경로를 지정해야 합니다.

-device 연결된 장치의 **ios_simulator**, 일련 번호(Android) 또는 핸들(iOS)을 지정합니다. iOS에서는 이 매개 변수가 필수적입니다. Android에서는 둘 이상의 Android 장치 또는 에뮬레이터가 컴퓨터에 연결되어 실행 중인 경우에만 이 매개 변수를 지정하면 됩니다. 지정한 장치가 연결되지 않은 경우 ADT에서 종료 코드 14: 장치 오류(Android) 또는 잘못된 장치 지정(iOS)을 반환합니다. 여러 개의 장치 또는 에뮬레이터가 연결되어 있는데 장치를 지정하지 않으면 ADT에서 종료 코드 2: 사용 오류를 반환합니다.

참고: iOS 장치에 IPA 파일을 직접 설치하는 기능은 AIR 3.4 이상에서 지원되며 iTunes 10.5.0 이상을 필요로 합니다.

adt -devices 명령(AIR 3.4 이상에서 사용 가능)을 사용하여 연결된 장치의 핸들 또는 일련 번호를 확인하십시오. iOS에서는 장치 UUID가 아니라 핸들을 사용합니다. 자세한 내용은 160페이지의 “ADT devices 명령”을 참조하십시오.

또한 Android에서는 Android ADB 도구를 사용하여 연결된 장치 및 실행 중인 에뮬레이터의 일련 번호를 나열할 수 있습니다.

```
adb devices
```

-package 설치할 패키지의 파일 이름입니다. iOS에서는 IPA 파일이어야 합니다. Android에서는 APK 패키지여야 합니다. 지정된 패키지가 이미 설치되어 있으면 ADT에서 오류 코드 14: 장치 오류를 반환합니다.

ADT appVersion 명령

-appVersion 명령은 장치 또는 에뮬레이터에 설치된 응용 프로그램의 버전을 보고합니다. 이 명령은 다음 구문을 사용합니다.

```
adt -appVersion -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform 장치 플랫폼의 이름입니다. **ios** 또는 **android**를 지정합니다.

-platformsdk 대상 장치의 플랫폼 SDK에 대한 경로입니다.

- Android - AIR 2.6 이상의 SDK에는 관련 ADT 명령을 구현하는 데 필요한 Android SDK의 도구가 포함되어 있습니다. 다른 버전의 Android SDK를 사용하려는 경우에만 이 값을 설정하십시오. 또한 AIR_ANDROID_SDK_HOME 환경 변수가 이미 설정된 경우에는 명령줄에 플랫폼 SDK 경로를 입력할 필요가 없습니다. 둘 다 설정되어 있으면 명령줄에 입력된 경로가 사용됩니다.
- iOS - AIR SDK가 전용 iOS SDK와 함께 제공됩니다. -platformsdk 옵션을 사용하면 외부 SDK를 사용하여 응용 프로그램을 패키지화할 수 있으므로 전용 iOS SDK만 사용해야 하는 문제가 없습니다. 예를 들어 최신 iOS SDK를 사용하여 확장을 빌드한 경우 응용 프로그램을 패키지화할 때 해당 SDK를 지정할 수 있습니다. 또한 iOS 시뮬레이터와 함께 ADT를 사용하는 경우에는 항상 -platformsdk 옵션을 포함하여 iOS 시뮬레이터 SDK의 경로를 지정해야 합니다.

-deviceios_simulator 또는 장치의 일련 번호를 지정합니다. 컴퓨터에 여러 개의 Android 장치 또는 에뮬레이터가 연결되어 실행 중인 경우에만 장치를 지정하면 됩니다. 지정된 장치가 연결되어 있지 않으면 ADT에서 종료 코드 14: 장치 오류를 반환합니다. 여러 개의 장치 또는 에뮬레이터가 연결되어 있는데 장치를 지정하지 않으면 ADT에서 종료 코드 2: 사용 오류를 반환합니다.

Android에서는 Android ADB 도구를 사용하여 연결된 장치 및 실행 중인 에뮬레이터의 일련 번호를 나열할 수 있습니다.

```
adb devices
```

-appid 설치된 응용 프로그램의 AIR 응용 프로그램 ID입니다. 지정된 ID의 응용 프로그램이 장치에 설치되어 있지 않으면 ADT에서 종료 코드 14: 장치 오류를 반환합니다.

ADT launchApp 명령

-launchApp 명령은 장치 또는 에뮬레이터에서 설치된 응용 프로그램을 실행합니다. 이 명령은 다음 구문을 사용합니다.

```
adt -launchApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform 장치 플랫폼의 이름입니다. **ios** 또는 **android**를 지정합니다.

-platformsdk 대상 장치의 플랫폼 SDK에 대한 경로입니다.

- Android - AIR 2.6 이상의 SDK에는 관련 ADT 명령을 구현하는 데 필요한 Android SDK의 도구가 포함되어 있습니다. 다른 버전의 Android SDK를 사용하려는 경우에만 이 값을 설정하십시오. 또한 AIR_ANDROID_SDK_HOME 환경 변수가 이미 설정된 경우에는 명령줄에 플랫폼 SDK 경로를 입력할 필요가 없습니다. 둘 다 설정되어 있으면 명령줄에 입력된 경로가 사용됩니다.
- iOS - AIR SDK가 전용 iOS SDK와 함께 제공됩니다. -platformsdk 옵션을 사용하면 외부 SDK를 사용하여 응용 프로그램을 패키지화할 수 있으므로 전용 iOS SDK만 사용해야 하는 문제가 없습니다. 예를 들어 최신 iOS SDK를 사용하여 확장을 빌드한 경우 응용 프로그램을 패키지화할 때 해당 SDK를 지정할 수 있습니다. 또한 iOS 시뮬레이터와 함께 ADT를 사용하는 경우에는 항상 -platformsdk 옵션을 포함하여 iOS 시뮬레이터 SDK의 경로를 지정해야 합니다.

-deviceios_simulator 또는 장치의 일련 번호를 지정합니다. 컴퓨터에 여러 개의 Android 장치 또는 에뮬레이터가 연결되어 실행 중인 경우에만 장치를 지정하면 됩니다. 지정된 장치가 연결되어 있지 않으면 ADT에서 종료 코드 14: 장치 오류를 반환합니다. 여러 개의 장치 또는 에뮬레이터가 연결되어 있는데 장치를 지정하지 않으면 ADT에서 종료 코드 2: 사용 오류를 반환합니다.

Android에서는 Android ADB 도구를 사용하여 연결된 장치 및 실행 중인 에뮬레이터의 일련 번호를 나열할 수 있습니다.

```
adb devices
```

-appid 설치된 응용 프로그램의 AIR 응용 프로그램 ID입니다. 지정된 ID의 응용 프로그램이 장치에 설치되어 있지 않으면 ADT에서 종료 코드 14: 장치 오류를 반환합니다.

ADT uninstallApp 명령

-uninstallApp 명령은 원격 장치 또는 에뮬레이터에서 설치된 응용 프로그램을 완전히 제거합니다. 이 명령은 다음 구문을 사용합니다.

```
adt -uninstallApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform 장치 플랫폼의 이름입니다. **ios** 또는 **android**를 지정합니다.

-platformsdk 대상 장치의 플랫폼 SDK에 대한 경로입니다.

- Android - AIR 2.6 이상의 SDK에는 관련 ADT 명령을 구현하는 데 필요한 Android SDK의 도구가 포함되어 있습니다. 다른 버전의 Android SDK를 사용하려는 경우에만 이 값을 설정하십시오. 또한 AIR_ANDROID_SDK_HOME 환경 변수가 이미 설정된 경우에는 명령줄에 플랫폼 SDK 경로를 입력할 필요가 없습니다. 둘 다 설정되어 있으면 명령줄에 입력된 경로가 사용됩니다.
- iOS - AIR SDK가 전용 iOS SDK와 함께 제공됩니다. -platformsdk 옵션을 사용하면 외부 SDK를 사용하여 응용 프로그램을 패키지화할 수 있으므로 전용 iOS SDK만 사용해야 하는 문제가 없습니다. 예를 들어 최신 iOS SDK를 사용하여 확장을 빌드한 경우 응용 프로그램을 패키지화할 때 해당 SDK를 지정할 수 있습니다. 또한 iOS 시뮬레이터와 함께 ADT를 사용하는 경우에는 항상 -platformsdk 옵션을 포함하여 iOS 시뮬레이터 SDK의 경로를 지정해야 합니다.

-deviceios_simulator 또는 장치의 일련 번호를 지정합니다. 컴퓨터에 여러 개의 Android 장치 또는 에뮬레이터가 연결되어 실행 중인 경우에만 장치를 지정하면 됩니다. 지정된 장치가 연결되어 있지 않으면 ADT에서 종료 코드 14: 장치 오류를 반환합니다. 여러 개의 장치 또는 에뮬레이터가 연결되어 있는데 장치를 지정하지 않으면 ADT에서 종료 코드 2: 사용 오류를 반환합니다.

Android에서는 Android ADB 도구를 사용하여 연결된 장치 및 실행 중인 에뮬레이터의 일련 번호를 나열할 수 있습니다.

```
adb devices
```

-appid 설치된 응용 프로그램의 AIR 응용 프로그램 ID입니다. 지정된 ID의 응용 프로그램이 장치에 설치되어 있지 않으면 ADT에서 종료 코드 14: 장치 오류를 반환합니다.

ADT installRuntime 명령

-installRuntime 명령은 AIR 런타임을 장치에 설치합니다.

먼저 기존 버전의 AIR 런타임을 제거한 후에야 이 명령을 사용하여 다시 설치할 수 있습니다.

이 명령은 다음 구문을 사용합니다.

```
adt -installRuntime -platform platformName -platformsdk path_to_sdk -device deviceID -package fileName
```

-platform 장치 플랫폼의 이름입니다. 현재 이 명령은 Android 플랫폼에서만 지원됩니다. **android**라는 이름을 사용하십시오.

-platformsdk 대상 장치의 플랫폼 SDK에 대한 경로입니다. 현재는 Android가 유일하게 지원되는 플랫폼 SDK입니다. AIR 2.6 이상의 SDK에는 관련 ADT 명령을 구현하는 데 필요한 Android SDK의 도구가 포함되어 있습니다. 다른 버전의 Android SDK를 사용하려는 경우에만 이 값을 설정하십시오. 또한 AIR_ANDROID_SDK_HOME 환경 변수가 이미 설정된 경우에는 명령줄에 플랫폼 SDK 경로를 입력할 필요가 없습니다. 둘 다 설정되어 있으면 명령줄에 입력된 경로가 사용됩니다.

-device 장치의 일련 번호입니다. 컴퓨터에 여러 개의 장치 또는 에뮬레이터가 연결되고 있고 실행 중일 때만 장치를 지정하면 됩니다. 지정된 장치가 연결되어 있지 않으면 ADT에서 종료 코드 14: 장치 오류를 반환합니다. 여러 개의 장치 또는 에뮬레이터가 연결되어 있는데 장치를 지정하지 않으면 ADT에서 종료 코드 2: 사용 오류를 반환합니다.

Android에서는 Android ADB 도구를 사용하여 연결된 장치 및 실행 중인 에뮬레이터의 일련 번호를 나열할 수 있습니다.

```
adb devices
```

-package 설치할 런타임의 파일 이름입니다. Android에서는 APK 패키지여야 합니다. 패키지가 지정되어 있지 않으면 AIR SDK에서 사용 가능한 것 중에서 장치 또는 에뮬레이터에 대해 적합한 런타임이 선택됩니다. 런타임이 이미 설치되어 있으면 ADT에서 오류 코드 14: 장치 오류를 반환합니다.

ADT runtimeVersion 명령

-runtimeVersion 명령은 장치 또는 에뮬레이터에 설치된 AIR 런타임의 버전을 보고합니다. 이 명령은 다음 구문을 사용합니다.

```
adt -runtimeVersion -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform 장치 플랫폼의 이름입니다. 현재 이 명령은 Android 플랫폼에서만 지원됩니다. **android**라는 이름을 사용하십시오.

-platformsdk 대상 장치의 플랫폼 SDK에 대한 경로입니다. 현재는 Android가 유일하게 지원되는 플랫폼 SDK입니다. AIR 2.6 이상의 SDK에는 관련 ADT 명령을 구현하는 데 필요한 Android SDK의 도구가 포함되어 있습니다. 다른 버전의 Android SDK를 사용하려는 경우에만 이 값을 설정하십시오. 또한 AIR_ANDROID_SDK_HOME 환경 변수가 이미 설정된 경우에는 명령줄에 플랫폼 SDK 경로를 입력할 필요가 없습니다. 둘 다 설정되어 있으면 명령줄에 입력된 경로가 사용됩니다.

-device 장치의 일련 번호입니다. 컴퓨터에 여러 개의 장치 또는 에뮬레이터가 연결되고 있고 실행 중일 때만 장치를 지정하면 됩니다. 런타임이 설치되어 있지 않거나 지정된 장치가 연결되어 있지 않으면 ADT에서 종료 코드 14: 장치 오류를 반환합니다. 여러 개의 장치 또는 에뮬레이터가 연결되어 있는데 장치를 지정하지 않으면 ADT에서 종료 코드 2: 사용 오류를 반환합니다.

Android에서는 Android ADB 도구를 사용하여 연결된 장치 및 실행 중인 에뮬레이터의 일련 번호를 나열할 수 있습니다.

```
adb devices
```

ADT uninstallRuntime 명령

-uninstallRuntime 명령은 장치 또는 에뮬레이터에서 AIR 런타임을 완전히 제거합니다. 이 명령은 다음 구문을 사용합니다.

```
adt -uninstallRuntime -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform 장치 플랫폼의 이름입니다. 현재 이 명령은 Android 플랫폼에서만 지원됩니다. **android**라는 이름을 사용하십시오.

-platformsdk 대상 장치의 플랫폼 SDK에 대한 경로입니다. 현재는 Android가 유일하게 지원되는 플랫폼 SDK입니다. AIR 2.6 이상의 SDK에는 관련 ADT 명령을 구현하는 데 필요한 Android SDK의 도구가 포함되어 있습니다. 다른 버전의 Android SDK를 사용하려는 경우에만 이 값을 설정하십시오. 또한 AIR_ANDROID_SDK_HOME 환경 변수가 이미 설정된 경우에는 명령줄에 플랫폼 SDK 경로를 입력할 필요가 없습니다. 둘 다 설정되어 있으면 명령줄에 입력된 경로가 사용됩니다.

-device 장치의 일련 번호입니다. 컴퓨터에 여러 개의 장치 또는 에뮬레이터가 연결되고 있고 실행 중일 때만 장치를 지정하면 됩니다. 지정된 장치가 연결되어 있지 않으면 ADT에서 종료 코드 14: 장치 오류를 반환합니다. 여러 개의 장치 또는 에뮬레이터가 연결되어 있는데 장치를 지정하지 않으면 ADT에서 종료 코드 2: 사용 오류를 반환합니다.

Android에서는 Android ADB 도구를 사용하여 연결된 장치 및 실행 중인 에뮬레이터의 일련 번호를 나열할 수 있습니다.

```
adb devices
```

ADT devices 명령

ADT -devices 명령은 현재 연결된 휴대 장치 및 에뮬레이터의 장치 ID를 표시합니다.

```
adt -devices -platform iOS|android
```

-platform 확인할 플랫폼의 이름입니다. android 또는 iOS를 지정합니다.

참고: iOS에 대해 이 명령을 사용하려면 iTunes 10.5.0 이상이 필요합니다.

ADT help 명령

ADT -help 명령은 명령줄 옵션에 대한 간결한 설명을 표시합니다.

```
adt -help
```

help 출력에 사용되는 심볼의 의미는 다음과 같습니다.

- <> - 각괄호 안의 항목은 직접 입력해야 하는 정보를 나타냅니다.
- () - 괄호 안의 항목은 help 명령 출력에서 한 묶음으로 취급되는 옵션을 나타냅니다.
- ALL_CAPS - 대문자로 표시된 항목은 따로 설명된 옵션 집합을 나타냅니다.
- | - OR 예를 들어 (A | B)는 항목 A 또는 항목 B를 의미합니다.
- ? - 0 또는 1 항목 뒤에 있는 물음표는 해당 항목이 선택 사항이며 사용되는 경우 하나의 인스턴스만 있을 수 있음을 나타냅니다.
- * - 0 이상 항목 뒤에 있는 별표는 해당 항목이 선택 사항이며 0개 이상의 인스턴스가 있을 수 있음을 나타냅니다.
- + - 1 이상. 항목 뒤에 있는 더하기 기호는 해당 항목이 필수 사항이며 여러 개의 인스턴스가 있을 수 있음을 나타냅니다.
- 심볼 없음 - 뒤에 아무런 심볼이 없는 항목은 필수 사항이며 하나의 인스턴스만 있을 수 있습니다.

ADT 옵션 집합

여러 ADT 명령이 공통된 옵션 집합을 공유합니다.

ADT 코드 서명 옵션

ADT는 JCA(Java Cryptography Architecture)를 사용하여 AIR 응용 프로그램에 서명하는 데 필요한 개인 키 및 인증서에 액세스합니다. 서명 옵션은 키 저장소 및 키 저장소 내의 개인 키와 인증서를 식별합니다.

키 저장소에는 개인 키 및 연관된 인증서 체인이 모두 들어 있어야 합니다. 서명 인증서가 컴퓨터에 있는 신뢰할 수 있는 인증서에 체인으로 연결되어 있는 경우 인증서의 공용 이름 필드에 있는 내용이 AIR 설치 대화 상자에 제작자 이름으로 표시됩니다.

ADT에서는 x509v3 표준(RFC3280)을 따르는 인증서를 필요로 하며 적절한 코드 서명 값을 사용하여 확장 키 사용 확장을 포함시킵니다. 인증서 내에 정의된 제약 조건을 준수하고 AIR 응용 프로그램에 서명할 일부 인증서 사용을 제외시킬 수 있습니다.

참고: ADT는 필요한 경우 인증서 해지 목록을 확인하고 타임스탬프를 가져오기 위해 Java 런타임 환경 프록시 설정을 사용하여 인터넷 리소스에 연결합니다. ADT 사용 시 이러한 인터넷 리소스에 연결하는 데 문제가 있어 네트워크에 특정 프록시 설정이 필요한 경우 JRE 프록시 설정을 구성해야 합니다.

AIR 서명 옵션 구문

서명 옵션은 다음 구문을 사용합니다(단일 명령줄임).

```
-alias aliasName
-storetype type
-keystore path
-storepass password1
-keypass password2
-providerName className
-tsa url
```

-alias 키 저장소에 있는 키의 별칭입니다. 키 저장소에 단 하나의 인증서만 들어 있을 때는 별칭을 지정할 필요가 없습니다. 별칭을 지정하지 않을 경우 ADT는 키 저장소의 첫 번째 키를 사용합니다.

모든 키 저장소 관리 응용 프로그램이 인증서에 대한 별칭 지정을 허용하는 것은 아닙니다. 예를 들어 Windows 시스템 키 저장소를 사용할 경우 인증서의 고유 이름을 별칭으로 사용합니다. 별칭을 확인할 수 있도록 Java Keytool 유틸리티를 사용하여 사용 가능한 인증서를 나열할 수 있습니다. 예를 들어 다음과 같은 명령을 실행합니다.

```
keytool -list -storetype Windows-MY
```

그러면 인증서에 대해 다음과 같은 출력이 생성됩니다.

```
CN=TestingCert,OU=QE,O=Adobe,C=US, PrivateKeyEntry,
Certificate fingerprint (MD5): 73:D5:21:E9:8A:28:0A:AB:FD:1D:11:EA:BB:A7:55:88
```

ADT 명령줄에서 이 인증서를 참조하려면 별칭을 다음과 같이 설정합니다.

```
CN=TestingCert,OU=QE,O=Adobe,C=US
```

Mac OS X에서 키체인의 인증서 별칭은 키체인 접근 응용 프로그램에 표시된 이름입니다.

-storetype 키 저장소의 유형으로 키 저장소 구현 시 결정됩니다. 대부분의 Java 설치 시 포함되는 기본 키 저장소 구현은 JKS 및 PKCS12 유형이 지원됩니다. Java 5.0에는 하드웨어 토큰에서 키 저장소에 액세스하기 위한 PKCS11 유형 및 Mac OS X 키체인에 액세스하기 위한 Keychain 유형에 대한 지원이 포함됩니다. Java 6.0에는 MSCAPI 유형에 대한 지원이 포함됩니다(Windows에서). 다른 JCA 공급자가 설치 및 구성되어 있는 경우 추가적인 키 저장소 유형을 사용할 수도 있습니다. 키 저장소 유형을 지정하지 않을 경우 기본 JCA 공급자에 대한 기본 유형이 사용됩니다.

저장소 유형	키 저장소 형식	최소 Java 버전
JKS	Java 키 저장소 파일(.keystore)	1.2
PKCS12	PKCS12 파일(.p12 또는 .pfx)	1.4
PKCS11	하드웨어 토큰	1.5
KeychainStore	Mac OS X 키체인	1.5
Windows-MY 또는 Windows-ROOT	MSCAPI	1.6

-keystore 파일 기반 저장소 유형을 위한 키 저장소 파일의 경로입니다.

-storepass 키 저장소에 액세스하는 데 필요한 암호입니다. 지정하지 않는 경우 ADT가 암호 입력을 요구합니다.

-keypass AIR 응용 프로그램 서명에 사용되는 개인 키에 액세스하는 데 필요한 암호입니다. 지정하지 않는 경우 ADT가 암호 입력을 요구합니다.

참고: 암호를 ADT 명령의 일부분으로 입력하는 경우 명령줄 작업 내역에 암호 문자가 저장됩니다. 따라서 인증서의 보안이 중요할 때는 -keypass 또는 -storepass 옵션을 사용하지 않는 것이 좋습니다. 또한 암호 옵션을 생략하면 같은 보안상의 이유로 인해 암호 프롬프트에 입력하는 문자가 표시되지 않습니다. 간단히 암호를 입력하고 Enter 키를 누르십시오.

-providerName 지정된 키 저장소 유형의 JCA 공급자입니다. 지정하지 않는 경우 ADT는 해당 유형의 키 저장소에 대한 기본 공급자를 사용합니다.

-tsa RFC3161 호환 타임스탬프 서버의 URL을 지정하여 디지털 서명에 타임스탬프를 포함시킵니다. URL을 지정하지 않으면 Geotrust가 제공한 기본 타임스탬프 서버가 사용됩니다. AIR 응용 프로그램의 서명에 타임스탬프가 포함되어 있을 경우 타임스탬프가 서명 시점에 인증서가 유효함을 확인한 것이므로 서명 인증서가 만료된 후에도 응용 프로그램을 설치할 수 있습니다.

ADT가 타임스탬프 서버에 연결되지 않으면 서명이 취소되고 패키지가 만들어지지 않습니다. 타임스탬프를 포함시키지 않으려면 -tsa none을 지정합니다. 그러나 타임스탬프 없이 패키지화된 AIR 응용 프로그램은 서명 인증서가 만료된 후에는 더 이상 설치할 수 없습니다.

참고: 다수의 서명 옵션은 Java Keytool 유틸리티의 같은 옵션과 동일합니다. Windows에서는 Keytool 유틸리티를 사용하여 키 저장소를 검사 및 관리할 수 있습니다. Mac OS X의 경우 Apple® 보안 유틸리티를 사용할 수도 있습니다.

-provisioning-profile Apple iOS 프로비저닝 파일입니다. iOS 응용 프로그램을 패키지화할 때만 필요합니다.

서명 옵션 예제

.p12 파일을 사용한 서명:

```
-storetype pkcs12 -keystore cert.p12
```

기본 Java 키 저장소를 사용한 서명:

```
-alias AIRcert -storetype jks
```

특정 Java 키 저장소를 사용한 서명:

```
-alias AIRcert -storetype jks -keystore certStore.keystore
```

Mac OS X 키체인을 사용한 서명:

```
-alias AIRcert -storetype KeychainStore -providerName Apple
```

Windows 시스템 키 저장소를 사용한 서명:

```
-alias cn=AIRCert -storeype Windows-MY
```

하드웨어 토큰을 사용한 서명(토큰을 사용하도록 Java를 구성하는 방법 및 올바른 providerName 값은 토큰 제조업체의 지침 참조):

```
-alias AIRCert -storetype pkcs11 -providerName tokenProviderName
```

타임스탬프를 포함하지 않는 서명:

```
-storetype pkcs12 -keystore cert.p12 -tsa none
```

파일 및 경로 옵션

파일 및 경로 옵션은 패키지에 포함된 모든 파일을 지정합니다. 파일 및 경로 옵션은 다음 구문을 사용합니다.

```
files_and_dirs -C dir files_and_dirs -e file_or_dir dir -extdir dir
```

files_and_dirs AIR 파일에서 패키지화할 파일 및 디렉토리입니다. 파일 및 디렉토리는 개수에 관계없이 지정할 수 있으며 공백으로 구분합니다. 디렉토리를 지정할 경우 해당 디렉토리 내의 숨겨진 파일을 제외한 모든 파일 및 하위 디렉토리가 패키지에 추가됩니다. 또한 직접 지정하거나 와일드 카드 또는 디렉토리 확장을 통해 응용 프로그램 설명자 파일을 지정할 경우 두 번째는 무시되고 패키지에 추가되지 않습니다. 지정된 파일 및 디렉토리는 현재 디렉토리에 있거나 하위 디렉토리 중 하나에 있어야 합니다. 현재 디렉토리를 변경하려면 -C 옵션을 사용합니다.

중요: –C 옵션 다음에 나오는 file_or_dir 인수에는 와일드 카드를 사용할 수 없습니다. 인수를 ADT에 전달하기 전에 명령 셸이 와일드 카드를 확장하므로, 이로 인해 ADT가 잘못된 위치에서 파일을 검색할 수 있기 때문입니다. 그러나 현재 디렉토리를 나타내는 도트 문자(".")는 계속 사용할 수 있습니다. 예를 들면 "-C assets .는 하위 디렉토리를 포함하여 에셋 디렉토리의 모든 항목을 응용 프로그램 패키지의 루트 레벨로 복사합니다.

-C dir files_and_dirs 응용 프로그램 패키지에 추가되는 후속 파일 및 디렉토리를 처리하기 전에 작업 디렉토리를 **dir**의 값으로 변경합니다(파일 및 디렉토리는 **files_and_dirs**에서 지정됨). 파일 또는 디렉토리가 응용 프로그램 패키지의 루트에 추가됩니다. –C 옵션을 여러 번 사용하여 파일 시스템의 여러 지점에 있는 파일을 포함시킬 수 있습니다. dir에 상대 경로를 지정할 경우 경로는 항상 원래 작업 디렉토리에 대한 상대 경로로 해석됩니다.

ADT는 패키지에 포함된 파일 및 디렉토리를 처리하므로 현재 디렉토리와 대상 파일 간의 상대 경로가 저장됩니다. 이 경로는 패키지가 설치될 때 응용 프로그램 디렉토리 구조로 확장됩니다. 따라서 -C release/bin lib/feature.swf를 지정하면 루트 응용 프로그램 폴더의 lib 하위 디렉토리에 release/bin/lib/feature.swf 파일이 저장됩니다.

-e file_or_dir dir 파일 또는 디렉토리를 지정된 패키지 디렉토리에 배치합니다. ANE 파일을 패키지화할 때는 이 옵션을 사용할 수 없습니다.

참고: 응용 프로그램 설명자 파일의 <content> 요소에서는 응용 프로그램 패키지 디렉토리 트리 내에서 기본 응용 프로그램 파일의 최종 위치를 지정해야 합니다.

-extdir dir*dir*의 값은 기본 확장(ANE 파일)을 검색할 디렉토리의 이름입니다. 절대 경로나 현재 디렉토리에 대한 상대 경로를 지정합니다. -extdir 옵션은 여러 번 지정할 수 있습니다.

지정한 디렉토리에는 응용 프로그램에 사용되는 기본 확장용 ANE 파일이 포함되어 있습니다. 이 디렉토리의 각 ANE 파일은 파일 이름 확장명이 .ane입니다. 하지만 .ane 파일 이름 확장자 앞에 있는 파일 이름은 응용 프로그램 설명자 파일의 extensionID 요소 값과 일치하지 않아도 됩니다.

예를 들어 -extdir ./extensions를 사용할 경우 extensions 디렉토리는 다음과 같을 수 있습니다.

```
extensions/
    extension1.ane
    extension2.ane
```

참고: -extdir 옵션의 사용 방법은 ADT 도구와 ADL 도구 간에 서로 다릅니다. ADL에서 이 옵션은 각각 패키지화되지 않은 ANE 파일을 포함하는 하위 디렉토리가 있는 디렉토리를 지정합니다. ADT에서 이 옵션은 ANE 파일이 포함된 디렉토리를 지정합니다.

디버거 연결 옵션

패키지의 대상이 apk-debug, ipa-debug 또는 ipa-debug-interpreter인 경우 연결 옵션을 사용하여 응용 프로그램에서 원격 디버거에 연결을 시도할지(일반적으로 wifi 디버깅에 사용), 아니면 원격 디버거에서 들어오는 연결을 수신 대기할지(일반적으로 USB 디버깅에 사용)를 지정할 수 있습니다. 디버거에 연결하려면 -connect 옵션을 사용하고, USB 연결을 통해 디버거로부터 연결을 수락하려면 -listen 옵션을 사용하십시오. 이 두 옵션은 상호 배타적이므로 함께 사용할 수 없습니다.

-connect 옵션은 다음 구문을 사용합니다.

```
-connect hostString
```

-connect 이 항목을 사용하면 응용 프로그램이 원격 디버거에 연결하려고 시도합니다.

hostString Flash 디버깅 도구 FDB를 실행하는 컴퓨터를 식별하는 문자열입니다. 지정하지 않을 경우 응용 프로그램은 패키지가 생성된 컴퓨터에서 실행 중인 디버거에 연결하려고 시도합니다. 호스트 문자열은 정규화된 컴퓨터 도메인 이름(**machinename.subgroup.example.com**) 또는 IP 주소(**192.168.4.122**)가 될 수 있습니다. 지정된(또는 기본) 컴퓨터를 찾을 수 없으면 런타임에서 유효한 호스트 이름을 요청하는 대화 상자를 표시합니다.

-listen 옵션은 다음 구문을 사용합니다.

```
-listen port
```

-listen 옵션이 사용되면 런타임이 원격 디버거로부터 연결을 기다립니다.

port (선택 사항) 수신 대기할 포트입니다. 기본적으로 런타임은 포트 7936에서 수신 대기합니다. -listen 옵션 사용에 대한 자세한 내용은 96페이지의 “USB를 통해 FDB를 사용하여 원격 디버깅”을 참조하십시오.

Android 응용 프로그램 프로파일링 옵션

패키지의 대상이 apk-profile일 때는 프로파일러 옵션을 사용하여 성능 및 메모리 프로파일링에 사용할 미리 로드된 SWF 파일을 지정할 수 있습니다. 프로파일러 옵션은 다음 구문을 사용합니다.

```
-preloadSWFPath directory
```

-preloadSWFPath 이 항목을 사용하면 응용 프로그램이 지정된 디렉토리에서 미리 로드 SWF를 찾으려고 시도합니다. 지정하지 않을 경우 ADT는 AIR SDK의 미리 로드 SWF 파일을 포함합니다.

directory 프로파일러 미리 로드 SWF 파일을 포함하는 디렉토리입니다.

기본 확장 옵션

기본 확장 옵션은 기본 확장용 ANE 파일을 패키지화하기 위한 옵션 및 파일을 지정합니다. -target 옵션이 ane인 ADT 패키지 명령에서 이러한 옵션을 사용합니다.

```
extension-descriptor -swc swcPath
   -platform platformName
  -platformoptions path/platform.xml
   FILE_OPTIONS
```

extension-descriptor 기본 확장의 설명자 파일입니다.

-swc 기본 확장의 ActionScript 코드 및 리소스를 포함하는 SWC 파일입니다.

-platform 이 ANE 파일이 지원하는 플랫폼의 이름입니다. 각각 고유한 FILE_OPTIONS가 있는 여러 -platform 옵션을 포함할 수 있습니다.

-platformoptions 플랫폼 옵션(platform.xml) 파일의 경로입니다. 이 파일을 사용하여 기본값이 아닌 링커 옵션, 공유 라이브러리, 그리고 확장에서 사용하는 타사 정적 라이브러리를 지정합니다. 자세한 내용 및 예제는 iOS 기본 라이브러리를 참조하십시오.

FILE_OPTIONS 기본 확장 패키지에 포함할 정적 라이브러리 등 패키지에 포함할 기본 플랫폼 파일을 식별합니다. 파일 옵션은 163페이지의 "파일 및 경로 옵션"에서 자세히 설명합니다. ANE 파일을 패키지화할 때는 -e 옵션을 사용할 수 없습니다.

기타 도움말 항목

기본 확장 패키지화

ADT 오류 메시지

다음 표에서는 ADT 프로그램에서 보고할 수 있는 오류 및 가능한 원인의 목록을 보여 줍니다.

응용 프로그램 설명자 유효성 검사 오류

오류 코드	설명	참고 사항
100	응용 프로그램 설명자를 파싱할 수 없습니다.	응용 프로그램 설명자 파일에 닫히지 않은 태그와 같은 XML 구문 오류가 있는지 확인합니다.
101	네임스페이스가 없습니다.	누락된 네임스페이스를 추가합니다.
102	네임스페이스가 잘못되었습니다.	네임스페이스의 맞춤법을 확인합니다.
103	예기치 못한 요소 또는 특성입니다.	잘못된 요소 및 특성을 제거합니다. 사용자 정의 값은 설명자 파일에 사용할 수 없습니다. 요소 및 특성 이름의 맞춤법을 확인합니다. 요소가 올바른 부모 요소 내에 배치되었는지, 특성이 올바른 요소에 사용되었는지 확인합니다.
104	요소 또는 특성이 없습니다.	필요한 요소 또는 특성을 추가합니다.
105	요소 또는 특성에 잘못된 값이 포함되어 있습니다.	잘못된 값을 수정합니다.
106	잘못된 윈도우 특성 조합입니다.	transparency = true 및 systemChrome = standard와 같은 일부 윈도우 설정은 함께 사용할 수 없습니다. 호환되지 않는 설정 중 하나를 변경합니다.

오류 코드	설명	참고 사항
107	윈도우 최소 크기가 윈도우 최대 크기보다 큽니다.	최소 또는 최대 크기 설정을 변경합니다.
108	이전 요소에서 이미 사용된 특성입니다.	
109	중복된 요소입니다.	중복된 요소를 제거합니다.
110	지정된 유형의 요소가 하나 이상 필요합니다.	누락된 요소를 추가합니다.
111	응용 프로그램 설명자에 나열된 프로파일 중에서 기본 확장을 지원하는 프로파일이 없습니다.	기본 확장을 지원하는 supportedProfies 목록에 프로파일을 추가합니다.
112	AIR 대상이 기본 확장을 지원하지 않습니다.	기본 확장을 지원하는 대상을 선택합니다.
113	<nativeLibrary> 및 <initializer>를 함께 제공해야 합니다.	기본 확장에 있는 모든 기본 라이브러리에 대해 이니셜라이저 함수를 지정해야 합니다.
114	<nativeLibrary>가 없는 <finalizer>가 발견되었습니다.	플랫폼에서 기본 라이브러리를 사용하지 않는 한 파이널라이저를 지정하지 않습니다.
115	기본 플랫폼은 기본 구현을 포함해서는 안 됩니다.	기본 플랫폼 요소에 기본 라이브러리를 지정하지 않습니다.
116	이 대상에 대해 브라우저 호출이 지원되지 않습니다.	지정된 패키지화 대상에 대해 <allowBrowserInvocation> 요소가 true일 수 없습니다.
117	기본 확장을 패키지화하려면 이 대상에 적어도 네임스페이스 n이 필요합니다.	응용 프로그램 설명자에서 AIR 네임스페이스를 지원되는 값으로 변경합니다.

네임스페이스, 요소, 특성 및 유효한 해당 값에 대한 자세한 내용은 183페이지의 “AIR 응용 프로그램 설명자 파일”을 참조하십시오.

응용 프로그램 아이콘 오류

오류 코드	설명	참고 사항
200	아이콘 파일을 열 수 없습니다.	파일이 지정된 경로에 있는지 확인합니다. 다른 응용 프로그램을 사용하여 파일을 열 수 있는지 확인합니다.
201	아이콘의 크기가 잘못되었습니다.	아이콘 크기(픽셀)는 XML 태그와 일치해야 합니다. 예를 들어 다음 응용 프로그램 설명자 요소가 있을 경우 <image32x32>icon.png</image32x32> icon.png의 이미지 크기는 정확히 32x32픽셀이어야 합니다.
202	아이콘 파일에 지원되지 않는 이미지 형식이 포함되어 있습니다.	PNG 형식만 지원됩니다. 응용 프로그램을 패키지화하기 전에 이미지를 다른 형식으로 변환합니다.

응용 프로그램 파일 오류

오류 코드	설명	참고 사항
300	파일이 없거나 파일을 열 수 없습니다.	명령줄에 지정된 파일을 찾을 수 없거나 열 수 없습니다.
301	응용 프로그램 설명자 파일이 없거나 파일을 열 수 없습니다.	응용 프로그램 설명자 파일이 지정된 경로에 없거나 파일을 열 수 없습니다.
302	루트 내용 파일이 패키지에 없습니다.	응용 프로그램 설명자의 <content> 요소에서 참조되는 SWF 또는 HTML 파일을 ADT 명령줄에 나열된 파일에 포함하여 패키지에 추가해야 합니다.
303	패키지에 아이콘 파일이 없습니다.	응용 프로그램 설명자에 지정된 아이콘 파일을 ADT 명령줄에 나열된 파일에 포함하여 패키지에 추가해야 합니다. 아이콘 파일은 자동으로 추가되지 않습니다.
304	초기 윈도우 내용이 잘못되었습니다.	응용 프로그램 설명자의 <content> 요소에서 참조되는 파일은 올바른 HTML 또는 SWF 파일로 인식되지 않습니다.
305	초기 윈도우 내용 SWF 버전이 네임스페이스 버전을 초과합니다.	응용 프로그램 설명자의 <content> 요소에서 참조되는 파일의 SWF 버전은 설명자 네임스페이스에 지정된 버전의 AIR에서 지원되지 않습니다. 예를 들어 SWF10(Flash Player 10) 파일을 AIR 1.1 응용 프로그램의 초기 내용으로 패키지화하려고 하면 이 오류가 발생합니다.
306	프로파일이 지원되지 않습니다.	응용 프로그램 설명자 파일에서 지정한 프로파일이 지원되지 않습니다. 216페이지의 "supportedProfiles"를 참조하십시오.
307	네임스페이스는 최소한 **nnn**이어야 합니다.	응용 프로그램에서 사용되는 기능에 대해 적절한 네임스페이스를 사용합니다(예: 2.0 네임스페이스).

기타 오류에 대한 종료 코드

종료 코드	설명	참고 사항
2	사용 오류입니다.	명령줄 인수에 오류가 없는지 확인합니다.
5	알 수 없는 오류입니다.	이 오류는 일반적인 오류 상황으로 설명할 수 없는 상황을 나타냅니다. ADT와 Java 런타임 환경이 호환되지 않거나, ADT 또는 JRE 설치가 손상되었거나, ADT 내에 프로그래밍 오류가 발생하는 등의 근본적인 원인입니다.
6	출력 디렉토리에 쓸 수 없습니다.	지정되거나 암시된 출력 디렉토리에 액세스할 수 있는지, 포함된 드라이브의 디스크 공간이 충분한지 확인합니다.
7	인증서에 액세스할 수 없습니다.	키 저장소에 대한 경로가 올바르게 지정되었는지 확인합니다. 키 저장소 내의 인증서에 액세스할 수 있는지 확인합니다. Java 1.6 Keytool 유틸리티를 사용하면 인증서 액세스 문제를 해결하는 데 도움이 됩니다.

종료 코드	설명	참고 사항
8	인증서가 잘못되었습니다.	인증서 파일의 형식이 잘못되었거나 인증서 파일이 수정, 만료 또는 해지되었습니다.
9	AIR 파일을 서명할 수 없습니다.	ADT에 전달된 서명 옵션을 확인합니다.
10	타임스탬프를 만들 수 없습니다.	ADT에서 타임스탬프 서버에 대한 연결을 설정할 수 없습니다. 프록시 서버를 통해 인터넷에 연결하는 경우 JRE 프록시 설정을 구성해야 할 수 있습니다.
11	인증서 만들기 오류가 발생했습니다.	서명을 만드는 데 사용된 명령줄 인수를 확인합니다.
12	입력이 잘못되었습니다.	명령줄에서 ADT에 전달된 파일 경로 및 기타 인수를 확인합니다.
13	장치 SDK가 누락되었습니다.	장치 SDK 구성을 확인합니다. ADT에서 지정된 명령을 실행하는 데 필요한 장치 SDK를 찾을 수 없습니다.
14	장치 오류	ADT에서 장치 제한 또는 문제로 인해 명령을 실행할 수 없습니다. 예를 들어 이 종료 코드는 실제로 설치되지 않은 응용 프로그램을 제거하려고 할 때 나타납니다.
15	장치가 없습니다.	장치가 연결되어 있고 켜져 있는지 또는 에뮬레이터가 실행 중인지 확인합니다.
16	GPL 구성 요소가 누락되었습니다.	현재 AIR SDK에 요청 작업을 수행하는 데 필요한 모든 구성 요소 중 일부가 포함되어 있지 않습니다.
17	장치 패키지화 도구가 실패했습니다.	필요한 운영 체제 구성 요소가 없으므로 패키지를 만들 수 없습니다.

Android 오류

종료 코드	설명	참고 사항
400	현재 Android sdk 버전이 특성을 지원하지 않습니다.	특성 이름을 올바르게 입력했는지 그리고 특성이 그 특성이 나타나는 요소에 대한 올바른 특성인지를 확인합니다. Android 2.2 이후에 도입된 특성의 경우 ADT 명령에서 -platformsdk 플래그를 설정해야 할 수도 있습니다.
401	현재 Android sdk 버전이 특성 값을 지원하지 않습니다.	특성 값을 올바르게 입력했는지 그리고 특성 값이 특성에 대한 올바른 값인지를 확인합니다. Android 2.2 이후에 도입된 특성 값의 경우 ADT 명령에서 -platformsdk 플래그를 설정해야 할 수도 있습니다.
402	현재 Android sdk 버전이 XML 태그를 지원하지 않습니다.	XML 태그 이름을 올바르게 입력했는지 그리고 XML 태그 이름이 올바른 Android 매니페스트 문서 요소인지를 확인합니다. Android 2.2 이후에 도입된 요소의 경우 ADT 명령에서 -platformsdk 플래그를 설정해야 할 수도 있습니다.

종료 코드	설명	참고 사항
403	Android 태그는 무시할 수 없습니다.	응용 프로그램이 AIR에서 사용되도록 예약된 Android 매니페스트 요소를 무시하려고 합니다. 69페이지의 "Android 설정"을 참조하십시오.
404	Android 특성은 무시할 수 없습니다.	응용 프로그램이 AIR에서 사용되도록 예약된 Android 매니페스트 특성을 무시하려고 합니다. 69페이지의 "Android 설정"을 참조하십시오.
405	Android 태그 %1은(는) manifestAdditions 태그에서 첫 번째 요소여야 합니다.	지정된 태그를 필요한 위치로 이동합니다.
406	android 태그 %2의 특성 %1에 대한 값 %3이(가) 잘못되었습니다.	특성에 유효한 값을 제공합니다.

ADT 환경 변수

ADT에서는 다음 환경 변수의 값을 읽습니다(해당 환경 변수가 설정되어 있는 경우).

AIR_ANDROID_SDK_HOME은 Android SDK의 루트 디렉토리(도구 폴더를 포함하고 있는 디렉토리)에 대한 경로를 지정합니다. AIR 2.6 이상의 SDK에는 관련 ADT 명령을 구현하는 데 필요한 Android SDK의 도구가 포함되어 있습니다. 다른 버전의 Android SDK를 사용하려는 경우에만 이 값을 설정하십시오. 이 변수가 설정되어 있으면 -platformsdk 옵션이 필요한 ADT 명령을 실행할 때 이 옵션을 지정하지 않아도 됩니다. 이 변수와 명령줄 옵션이 모두 설정되어 있으면 명령줄에 지정된 경로가 사용됩니다.

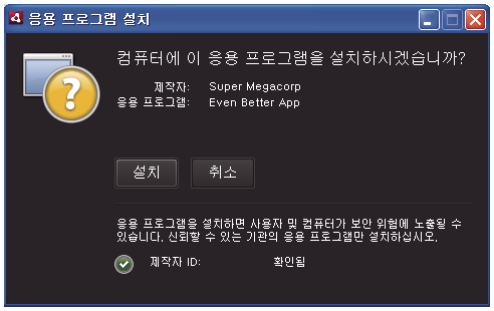
AIR_EXTENSION_PATH는 응용 프로그램에 필요한 기본 확장을 검색할 디렉토리 목록을 지정합니다. 이 디렉토리 목록은 ADT 명령줄에 지정된 기본 확장 디렉토리 이후에 순서대로 검색됩니다. 또한 ADL 명령에 이 환경 변수가 사용됩니다.

참고: 일부 컴퓨터 시스템에서는 이러한 환경 변수에 저장된 파일 시스템 경로에 있는 2바이트 문자가 잘못 해석될 수 있습니다. 그럴 경우에는 ADT를 실행하는 데 사용된 JRE이 UTF-8 문자 세트를 사용하도록 설정해 보십시오. 이 작업은 기본적으로 Mac 및 Linux에서 ADT를 실행하는 데 사용하는 스크립트에서 수행할 수 있습니다. Windows adt.bat 파일에서 또는 Java로부터 ADT를 직접 실행할 때 Java 명령줄에서 -Dfile.encoding=UTF-8 옵션을 지정하십시오.

13장: AIR 응용 프로그램 서명

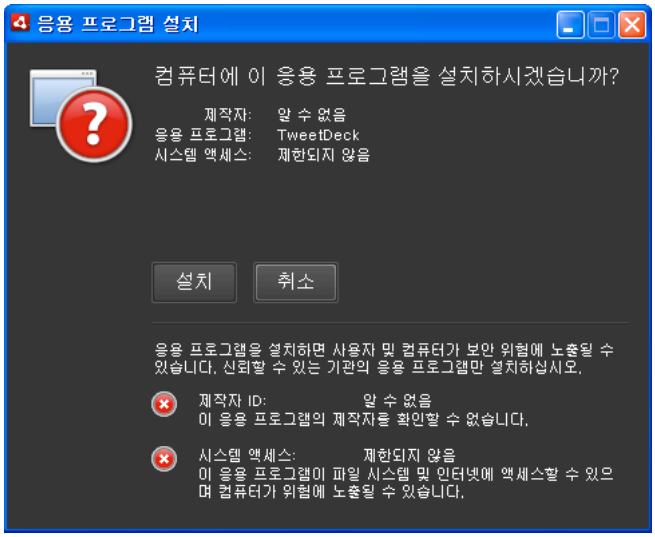
AIR 파일에 디지털 서명

공인된 CA(인증 기관)에서 발급한 인증서로 AIR 설치 파일에 디지털 서명하면 설치하는 응용 프로그램이 실수로 또는 악의적으로 변경되지 않았음이 확실히 보장되며 사용자가 서명자(제작자)의 신원을 확인할 수 있습니다. AIR는 AIR 응용 프로그램이 신뢰할 수 있는 인증서 또는 설치 컴퓨터에서 신뢰할 수 있는 인증서에 **체인으로 연결**된 인증서로 서명된 경우 설치하는 동안 제작자 이름을 표시합니다.



신뢰할 수 있는 인증서로 서명된 응용 프로그램의 설치 확인 대화 상자

자체 서명 인증서(또는 신뢰할 수 있는 인증서에 체인으로 연결되지 않은 인증서)로 응용 프로그램을 서명할 경우 사용자가 응용 프로그램 설치 시 더 큰 보안 위험을 감수해야 합니다. 설치 대화 상자에는 이러한 추가 위험에 대한 설명이 표시됩니다.



자체 서명 인증서로 서명된 응용 프로그램의 설치 확인 대화 상자

중요: 악의적인 사용자가 제작자의 서명 키 저장소 파일 또는 개인 키를 획득하게 될 경우 ID를 도용하여 AIR 파일을 위조할 수 있습니다.

코드 서명 인증서

코드 서명 인증서 사용과 관련된 보안 보증, 제한 사항 및 법적 규제는 인증 발급 기관에서 발행하는 CPS(Certificate Practice Statements) 및 구독자 계약서에 약술되어 있습니다. 현재 AIR 코드 서명 인증서를 발급하는 인증 기관의 계약서에 대한 자세한 내용은 다음을 참조하십시오.

ChosenSecurity(http://www.chosensecurity.com/products/tc_publisher_id_adobe_air.htm)

ChosenSecurity CPS(http://www.chosensecurity.com/resource_center/repository.htm)

GlobalSign(http://www.globalsign.com/code-signing/index.html)

GlobalSign CPS(http://www.globalsign.com/repository/index.htm)

Thawte CPS(http://www.thawte.com/cps/index.html)

VeriSign CPS(http://www.verisign.com/repository/CPS/)

VeriSign Subscriber's Agreement(https://www.verisign.com/repository/subscriber/SUBAGR.html)

AIR 코드 서명

AIR 파일에 서명하면 디지털 서명이 설치 파일에 포함됩니다. 서명에는 AIR 파일이 서명된 이후 변경되지 않았음을 확인하는 데 사용되는 패키지의 다이제스트와 제작자 ID를 확인하는 데 사용되는 서명 인증서에 대한 정보가 포함됩니다.

AIR는 운영 체제의 인증서 저장소를 통해 지원되는 PKI(공개 키 인프라)를 사용하여 인증서를 신뢰할 수 있는지 여부를 증명합니다. 제작자 정보를 확인하려면 AIR 응용 프로그램이 설치되어 있는 컴퓨터가 AIR 응용 프로그램에 서명하는 데 사용된 인증서를 직접 신뢰하거나 인증서를 신뢰할 수 있는 인증 기관에 연결하는 인증서 체인을 신뢰해야 합니다.

AIR 파일이 신뢰할 수 있는 루트 인증서(일반적으로 여기에는 자체 서명된 모든 인증서가 포함됨) 중 하나에 체인으로 연결되지 않은 인증서로 서명된 경우 제작자 정보를 확인할 수 없습니다. AIR는 AIR 패키지가 서명된 이후 변경되지 않았음을 확인할 수 있지만 해당 파일을 실제로 누가 만들고 서명했는지는 알 수 없습니다.

참고: 사용자는 자체 서명된 인증서를 신뢰하도록 선택할 수 있으며 이 경우 인증서로 서명된 AIR 응용 프로그램에서 인증서의 공용 이름 필드 값이 제작자 이름으로 표시됩니다. AIR에는 사용자가 인증서를 신뢰할 수 있는 것으로 지정할 수 있는 방법이 없습니다. 개인 키가 포함되지 않은 인증서를 사용자에게 개별적으로 제공해야 하며 사용자는 운영 체제에서 제공하는 메커니즘 중 하나나 적절한 도구를 사용하여 인증서를 시스템 인증서 저장소의 적절한 위치로 가져와야 합니다.

AIR 제작자 ID

중요: AIR 1.5.3부터는 제작자 ID가 더 이상 사용되지 않으며 더 이상 코드 서명 인증서를 기반으로 계산되지 않습니다. 새로운 응용 프로그램에서는 제작자 ID를 사용할 필요가 없으며 사용해서도 안 됩니다. 기존 응용 프로그램을 업데이트하는 경우 응용 프로그램 설명자 파일에서 원래 제작자 ID를 지정해야 합니다.

AIR 1.5.3 이전에는 AIR 응용 프로그램 설치 프로그램이 AIR 파일 설치 중에 제작자 ID를 생성했습니다. 이 ID는 AIR 파일에 서명하는 데 사용된 인증서에 고유한 ID였습니다. 여러 AIR 응용 프로그램에 대해 동일한 인증서를 다시 사용하는 경우 해당 응용 프로그램은 모두 동일한 제작자 ID를 받았습니다. 다른 인증서로 응용 프로그램 업데이트에 서명하거나 원래 인증서의 인스턴스가 갱신되는 경우에도 제작자 ID가 변경되었습니다.

AIR 1.5.3 이상에서는 AIR에서 제작자 ID를 할당하지 않습니다. AIR 1.5.3을 사용하여 제작된 응용 프로그램은 응용 프로그램 설명자에서 제작자 ID 문자열을 지정할 수 있습니다. 원래 1.5.3 이전 버전의 AIR에 대해 제작된 응용 프로그램의 업데이트를 제작하는 경우에만 제작자 ID를 지정해야 합니다. 응용 프로그램 설명자에서 원래 ID를 지정하지 않으면 새로운 AIR 패키지가 기존 응용 프로그램의 업데이트로 간주되지 않습니다.

원래 제작자 ID를 확인하려면 원래 응용 프로그램이 설치된 META-INF/AIR 하위 디렉토리에서 publisherid 파일을 찾아보십시오. 이 파일 내에 들어 있는 문자열이 제작자 ID입니다. 제작자 ID를 수동으로 지정하려면 응용 프로그램 설명자가 응용 프로그램 설명자 파일의 네임스페이스 선언에서 AIR 1.5.3 런타임 이상을 지정해야 합니다.

제작자 ID(있을 경우)는 다음과 같은 용도로 사용됩니다.

- 암호화된 로컬 저장소를 위한 암호화 키의 일부
- 응용 프로그램 저장소 디렉토리 경로의 일부
- 로컬 연결을 위한 연결 문자열의 일부
- AIR 인 브라우저(in-browser) API를 통해 응용 프로그램을 호출하는 데 사용되는 ID 문자열의 일부
- OSID의 일부(사용자 정의 설치/제거 프로그램을 만들 때 사용됨)

제작자 ID가 변경되면 해당 ID를 사용하는 모든 AIR 기능의 비헤이비어도 변경됩니다. 예를 들어 기존 암호화된 로컬 저장소에 포함된 데이터에 더 이상 액세스할 수 없으며, 응용 프로그램에 대한 로컬 연결을 만드는 모든 Flash 또는 AIR 인스턴스는 연결 문자열에 새로운 ID를 사용해야 합니다. AIR 1.5.3 이상에서는 설치된 응용 프로그램의 제작자 ID를 변경할 수 없습니다. AIR 패키지를 제작할 때 다른 제작자 ID를 사용하면 설치 프로그램이 새 패키지를 업데이트가 아니라 다른 응용 프로그램으로 간주합니다.

인증서 형식

AIR 서명 도구는 JCA(Java Cryptography Architecture)를 통해 액세스할 수 있는 모든 키 저장소를 허용합니다. 여기에는 PKCS12 형식 파일(일반적으로 .pfx 또는 .p12 파일 확장명 사용)과 같은 파일 기반 키 저장소, Java .keystore 파일, PKCS11 하드웨어 키 저장소 및 시스템 키 저장소가 포함됩니다. ADT에서 액세스할 수 있는 키 저장소 형식은 ADT를 실행하는 데 사용된 Java 런타임의 버전과 구성에 따라 달라집니다. PKCS11 하드웨어 토큰과 같은 일부 키 저장소 유형에 액세스하려면 추가 소프트웨어 드라이버와 JCA 플러그인을 설치 및 구성해야 할 수 있습니다.

AIR 파일에 서명하기 위해 대부분의 기존 코드 서명 인증서를 사용하거나 AIR 응용 프로그램에 서명하기 위해 명시적으로 발급된 새 인증서를 얻을 수 있습니다. 예를 들어 다음과 같은 유형의 VeriSign, Thawte, GlobalSign 또는 ChosenSecurity 인증서 중 하나를 사용할 수 있습니다.

- ChosenSecurity:
  - TC Publisher ID for Adobe AIR
- GlobalSign
  - ObjectSign Code Signing Certificate
- Thawte:
  - AIR Developer Certificate
  - Apple Developer Certificate
  - JavaSoft Developer Certificate
  - Microsoft Authenticode Certificate
- VeriSign:
  - Adobe AIR Digital ID
  - Microsoft Authenticode Digital ID
  - Sun Java Signing Digital ID

참고: 코드 서명용으로 인증서를 만들어야 합니다. SSL 또는 기타 유형의 인증서를 AIR 파일에 서명하는 데 사용할 수 없습니다.

타임스탬프

AIR 파일에 서명하면 패키지 도구가 독립적으로 확인 가능한 서명 날짜와 시간을 가져오기 위해 타임스탬프 기관의 서버에 쿼리를 보냅니다. 가져온 타임스탬프는 AIR 파일에 포함됩니다. 서명 인증서가 서명 당시 유효하기만 하면 인증서가 만료된 후에도 AIR 파일을 설치할 수 있습니다. 반대로 타임스탬프를 가져오지 못하면 인증서가 만료되거나 해지될 경우 AIR 파일을 더 이상 설치할 수 없게 됩니다.

기본적으로 AIR 패키지 도구는 타임스탬프를 가져옵니다. 하지만 타임스탬프 서비스를 사용할 수 없을 때 응용 프로그램을 패키지화할 수 있도록 타임스탬프 기능을 해제할 수 있습니다. 공개적으로 배포되는 모든 AIR 파일에는 타임스탬프가 포함되는 것이 좋습니다.

AIR 패키지 도구에서 사용되는 기본 타임스탬프 기관은 Geotrust입니다.

인증서 얻기

인증서를 얻으려면 일반적으로 인증 기관 웹 사이트를 방문하여 해당 회사의 구매 프로세스를 완료해야 합니다. AIR 도구에 필요한 키 저장소 파일을 생성하는 데 사용되는 도구는 구매한 인증서의 유형, 수신 컴퓨터에서 인증서가 저장되는 방법 및 인증서를 얻는 데 사용한 브라우저(일부 경우)에 따라 달라집니다. 예를 들어 Thawte에서 Adobe Developer Certificate를 얻어 내보내려면 Mozilla Firefox를 사용해야 합니다. 그러면 Firefox 사용자 인터페이스에서 직접 인증서를 .p12 또는 .pfx 파일로 내보낼 수 있습니다.

참고: Java 버전 1.5 이상의 경우 PKCS12 인증서 파일을 보호하는 데 사용되는 암호에 상위 ASCII 문자를 사용할 수 없습니다. Java는 AIR 개발 도구에서 서명된 AIR 패키지를 만드는 데 사용됩니다. 인증서를 .p12 또는 .pfx 파일로 내보낼 경우 암호에 일반 ASCII 문자만 사용하십시오.

AIR 설치 파일을 패키지화하는 데 사용된 ADT(Air Development Tool)를 사용하여 자체 서명된 인증서를 생성할 수 있습니다. 일부 타사 도구도 사용할 수 있습니다.

자체 서명된 인증서를 생성하는 방법 및 AIR 파일 서명 방법에 대한 자세한 내용은 149페이지의 “ADT(AIR Developer Tool)”를 참조하십시오. Flash Builder, Dreamweaver 및 Flash용 AIR 업데이트를 사용하여 AIR 파일을 내보내고 서명할 수도 있습니다.

다음 예에서는 Thawte 인증 기관으로부터 AIR Developer Certificate를 얻는 방법과 ADT에 사용하기 위해 해당 인증서를 준비하는 방법을 설명합니다.

예제: Thawte로부터 AIR Developer Certificate 얻기

참고: 이 예는 사용할 코드 서명 인증서를 얻고 준비하는 많은 방법 중 하나를 보여 줄 뿐입니다. 인증 기관별로 자체적인 정책과 절차가 있습니다.

Thawte 웹 사이트에서 AIR Developer Certificate를 구매하려면 Mozilla Firefox 브라우저를 사용해야 합니다. 인증서의 개인 키는 브라우저의 키 저장소에 저장됩니다. Firefox 키 저장소가 마스터 암호로 보호되는지 그리고 컴퓨터 자체가 물리적으로 안전한지 확인하십시오. 구매 프로세스가 완료되면 브라우저 키 저장소에서 인증서와 개인 키를 내보내고 제거할 수 있습니다.

인증서 등록 프로세스의 일부로 개인/공개 키 쌍이 생성됩니다. 개인 키는 Firefox 키 저장소 내에 자동으로 저장됩니다. Thawte 웹 사이트에서 인증서를 요청하거나 검색할 때 항상 같은 컴퓨터와 브라우저를 사용해야 합니다.

1. Thawte 웹 사이트를 방문하여 Product 페이지의 Code Signing Certificates로 이동합니다.
2. Code Signing Certificates 목록에서 [Adobe AIR Developer Certificate]를 선택합니다.
3. 세 단계로 구성된 등록 프로세스를 완료합니다. 조직 및 연락처 정보를 제공해야 합니다. 그러면 Thawte에서 ID 확인 프로세스를 수행하고 추가 정보를 요청할 수 있습니다. 확인이 완료되면 Thawte에서 인증서를 검색하는 방법에 대한 지침을 전자 메일로 보냅니다.

   참고: 필요한 설명서 유형에 대한 자세한 내용은 https://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/enroll_codesign_eng.pdf를 참조하십시오.

4 Thawte 사이트에서 발급된 인증서를 검색합니다. 인증서는 자동으로 Firefox 키 저장소에 저장됩니다.

5 다음 단계를 사용하여 Firefox 키 저장소에서 개인 키 및 인증서가 포함된 키 저장소 파일을 내보냅니다.

참고: Firefox에서 개인 키/인증서를 내보낼 때 해당 파일은 ADT, Flex, Flash 및 Dreamweaver에서 사용할 수 있는 .p12(pfx) 형식으로 내보내집니다.

a [Firefox 인증서 관리자] 대화 상자를 엽니다.

b Windows의 경우: [도구] -> [옵션] -> [고급] -> [암호화] -> [인증서 보기]를 엽니다.

c Mac OS의 경우: [Firefox] -> [환경 설정] -> [고급] -> [암호화] -> [인증서 보기]를 엽니다.

d Linux의 경우: [편집] -> [환경 설정] -> [고급] -> [암호화] -> [인증서 보기]를 엽니다.

e 인증서 목록에서 [AIR Code Signing Certificate]를 선택하고 **[백업]** 버튼을 클릭합니다.

f 파일 이름과 키 저장소 파일을 내보낼 위치를 입력하고 **저장**을 클릭합니다.

g Firefox 마스터 암호를 사용 중인 경우 파일을 내보내려면 소프트웨어 보안 장치의 암호를 입력하라는 메시지가 표시됩니다. 이 암호는 Firefox에서만 사용됩니다.

h **인증서 백업 암호 선택** 대화 상자에서 키 저장소 파일의 암호를 만듭니다.

중요: 이 암호는 키 저장소 파일을 보호하며 AIR 응용 프로그램 서명에 파일을 사용할 때 필요합니다. 보안 암호를 선택해야 합니다.

i [확인]을 클릭합니다. 백업 암호가 만들어졌다는 메시지가 표시됩니다. 개인 키와 인증서가 포함된 키 저장소 파일은 .p12 파일 확장명(PKCS12 형식)으로 저장됩니다.

6 내보낸 키 저장소 파일을 ADT, Flash Builder, Flash Professional 또는 Dreamweaver에서 사용합니다. AIR 응용 프로그램에 서명할 때마다 파일에 대해 생성된 암호가 필요합니다.

중요: 개인 키와 인증서는 계속 Firefox 키 저장소 내에 저장되어 있습니다. 이로 인해 인증서 파일의 추가 복사본을 내보낼 수 있지만 인증서와 개인 키의 보안을 유지하기 위해 보호해야 하는 다른 액세스 지점도 노출됩니다.

인증서 변경

일부 경우, AIR 응용 프로그램의 업데이트에 서명하는 데 사용하는 인증서를 변경해야 합니다. 이러한 경우는 다음과 같습니다.

- 원래 서명 인증서를 갱신하는 경우
- 자체 서명된 인증서에서 인증 기관에서 발급한 인증서로 업그레이드하는 경우
- 만료 예정인 자체 서명된 인증서에서 다른 인증서로 변경하는 경우
- CI(Corporate Identity)가 변경되는 등의 이유로 인해 한 상업용 인증서에서 다른 상업용 인증서로 변경하는 경우

AIR에서 AIR 파일을 업데이트로 인식하도록 하려면 원래 AIR 파일과 업데이트 AIR 파일에 동일한 인증서로 서명하거나 업데이트에 인증서 마이그레이션 서명을 적용해야 합니다. 마이그레이션 서명은 원래 인증서를 사용하여 업데이트 AIR 패키지에 적용되는 2차 서명입니다. 마이그레이션 서명은 원래 인증서를 사용하여 서명자가 응용 프로그램의 원래 제작자임을 입증합니다.

마이그레이션 서명이 적용된 AIR 파일이 설치된 후에는 새 인증서가 기본 인증서가 됩니다. 이후 업데이트에는 마이그레이션 서명이 필요하지 않습니다. 그러나 업데이트를 건너뛰는 사용자도 수용할 수 있도록 가능한 한 오랫동안 마이그레이션 서명을 적용해야 합니다.

중요: 인증서를 변경하고, 원래 인증서가 만료되기 전에 원래 인증서를 사용하여 업데이트에 마이그레이션 서명을 적용해야 합니다. 그렇지 않으면 사용자가 새 버전을 설치하기 전에 기존 버전의 응용 프로그램을 제거해야 합니다. AIR 1.5.3 이상에서는 만료 후 365일의 유예 기간 내에 만료된 인증서를 사용하여 마이그레이션 서명을 적용할 수 있습니다. 하지만 만료된 인증서를 사용하여 기본 응용 프로그램 서명을 적용할 수는 없습니다.

인증서를 변경하려면

1 응용 프로그램의 업데이트를 만듭니다.

2 **새** 인증서를 사용하여 업데이트 AIR 파일을 패키지화하고 서명합니다.

3 **원래** 인증서(ADT -migrate 명령 사용)를 사용하여 다시 AIR 파일에 서명합니다.

다른 측면에서 보면 마이그레이션 서명이 있는 AIR 파일은 정상 AIR 파일입니다. 원래 버전이 없는 시스템에 응용 프로그램을 설치한 경우 AIR는 일반적인 방식으로 새 버전을 설치합니다.

참고: AIR 1.5.3 이전에는 갱신된 인증서로 AIR 응용 프로그램에 서명할 때 마이그레이션 서명이 필요하지 않은 경우도 있었습니다. AIR 1.5.3 버전부터는 갱신된 인증서를 사용할 경우 항상 마이그레이션 서명이 필요합니다.

마이그레이션 서명을 적용하려면 179페이지의 "업데이트된 버전의 AIR 응용 프로그램에 서명"에 설명된 대로 156페이지의 "ADT migrate 명령"을 사용합니다.

참고: 기본 확장이 포함된 AIR 데스크톱 응용 프로그램은 .air 파일이 아닌 기본 설치 프로그램으로 패키지화되기 때문에 이 AIR 데스크톱 응용 프로그램에는 ADT migrate 명령을 사용할 수 없습니다. 기본 확장이 포함된 AIR 데스크톱 응용 프로그램의 인증서를 변경하려면 150페이지의 "ADT package 명령"을 -migrate 플래그와 함께 사용하여 응용 프로그램을 패키지화합니다.

응용 프로그램 ID 변경

AIR 1.5.3 이전에는 마이그레이션 서명으로 서명된 업데이트가 설치될 경우 AIR 응용 프로그램의 ID가 변경되었습니다. 응용 프로그램의 ID를 변경하면 다음과 같은 몇 가지 부정적인 영향을 줍니다.

- 새 응용 프로그램 버전이 암호화된 기존 로컬 저장소의 데이터에 액세스할 수 없습니다.
- 응용 프로그램 저장소 디렉토리의 위치가 변경됩니다. 이전 위치의 데이터가 새 디렉토리로 복사되지 않습니다. 하지만 새 응용 프로그램은 이전 제작자 ID를 기반으로 원래 디렉토리를 찾을 수 있습니다.
- 응용 프로그램이 이전 제작자 ID를 사용하여 로컬 연결을 더 이상 열 수 없습니다.
- 웹 페이지에서 응용 프로그램에 액세스하는 데 사용되는 ID 문자열이 변경됩니다.
- 응용 프로그램의 OSID가 변경됩니다. OSID는 사용자 정의 설치/제거 프로그램을 작성할 때 사용됩니다.

AIR 1.5.3 이상에서 업데이트를 제작하는 경우에는 응용 프로그램 ID를 변경할 수 없습니다. 업데이트 AIR 파일의 응용 프로그램 설명자에 원래 응용 프로그램 및 제작자 ID가 지정되어야 합니다. 그렇지 않으면 새 패키지가 업데이트로 인식되지 않습니다.

참고: AIR 1.5.3 이상을 사용하여 새 AIR 응용 프로그램을 제작하는 경우에는 제작자 ID를 지정하면 안 됩니다.

용어

이 단원에서는 공개 배포를 위해 응용 프로그램에 서명하는 방법을 결정할 때 이해해야 할 핵심 용어 중 몇 가지에 대한 용어 사전을 제공합니다.

용어	설명
CA(인증 기관)	신뢰할 수 있는 타사 역할을 하며 궁극적으로 공개 키 소유자의 ID를 인증하는 공개 키 인프라 네트워크의 주체입니다. CA는 일반적으로 자체 개인 키로 서명된 디지털 인증서를 발급하여 인증서 소유자의 ID를 확인했음을 증명합니다.
CPS(Certificate Practice Statement)	인증서를 발급하고 확인하는 인증 기관의 사용 약관과 정책을 설명합니다. CPS는 CA와 해당 구독자 및 관련 당사자 간 계약의 일부입니다. 또한 ID 확인에 대한 정책과 해당 기관에서 제공하는 인증서의 보증 수준에 대해 간략하게 설명합니다.
CRL(인증서 해지 목록)	이미 해지되었으므로 더 이상 신뢰할 수 없게 된 인증서 목록입니다. AIR는 AIR 응용 프로그램이 서명될 때 CRL을 확인하고 타임스탬프가 없으면 응용 프로그램을 설치할 때 다시 CRL을 확인합니다.
인증서 체인	인증서 체인은 체인의 각 인증서가 다음 인증서에 의해 서명된 인증서의 시퀀스입니다.
디지털 인증서	소유자의 ID, 소유자의 공개 키 및 인증서 자체의 ID에 대한 정보가 포함된 디지털 문서입니다. 인증 기관에서 발급한 인증서 자체는 발급 주체 CA에 속하는 인증서에 의해 서명됩니다.

용어	설명
디지털 서명	공개-개인 키 쌍의 절반인 공개 키로만 해독할 수 있는 암호화된 메시지 또는 다이제스트입니다. PKI에서 디지털 서명에는 결국은 인증 기관에서 추적할 수 있는 하나 이상의 디지털 인증서가 포함됩니다. 메시지 또는 컴퓨터 파일에 서명한 후 변경되지 않았음(사용된 암호화 알고리즘에서 제공하는 보증 제한 내에서)을 검증하는 데 사용될 수 있습니다. 이 경우 발급 주체 인증 기관, 서명자의 ID를 신뢰한다고 가정합니다.
키 저장소	디지털 인증과 경우에 따라서는 관련 개인 키를 포함하는 데이터베이스입니다.
JCA(Java Cryptography Architecture)	키 저장소를 관리하고 액세스하는 확장 가능한 아키텍처입니다. 자세한 내용은 Java Cryptography Architecture Reference Guide를 참조하십시오.
PKCS #11	RSA Laboratories의 암호화 토큰 인터페이스 표준입니다. 하드웨어 토큰 기반 키 저장소입니다.
PKCS #12	RSA Laboratories의 개인 정보 교환 구문 표준입니다. 일반적으로 개인 키 및 연결된 디지털 인증서가 포함된 파일 기반 키 저장소입니다.
개인 키	두 부분으로 구성된 공개-개인 키 비대칭 암호화 시스템 중 절반인 개인 부분입니다. 개인 키는 비밀로 보관되어야 하며 네트워크를 통해 전송하면 안 됩니다. 디지털로 서명된 메시지는 서명자에 의해 개인 키로 암호화됩니다.
공개 키	두 부분으로 구성된 공개-개인 키 비대칭 암호화 시스템 중 절반인 공개 부분입니다. 공개 키는 공개적으로 사용할 수 있으며 개인 키로 암호화된 메시지를 해독하는 데 사용됩니다.
PKI(공개 키 인프라)	인증 기관이 공개 키 소유주의 ID에 증명하는 신뢰 시스템입니다. 네트워크의 클라이언트는 디지털 메시지 또는 파일 서명자의 ID를 확인하기 위해 신뢰할 수 있는 CA에서 발급한 디지털 인증서에 의존합니다.
타임스탬프	이벤트가 발생한 날짜 및 시간이 포함된 디지털로 서명된 데이터입니다. ADT는 AIR 패키지의 RFC 3161 호환 시간 서버에서 타임스탬프를 포함할 수 있습니다. 타임스탬프가 있는 경우 AIR는 타임스탬프를 사용하여 서명 시간의 인증서 유효성을 설정합니다. 이를 통해 AIR 응용 프로그램의 서명 인증서가 만료된 이후 해당 응용 프로그램이 설치될 수 있습니다.
타임스탬프 기관	타임스탬프를 발급하는 기관입니다. AIR가 인식할 수 있도록 타임스탬프는 RFC 3161을 따라야 하고 타임스탬프 서명은 설치 컴퓨터에서 신뢰할 수 있는 인증 기관에 연결되어야 합니다.

iOS 인증서

Apple에서 발급한 코드 서명 인증서를 사용하여 iOS 응용 프로그램을 서명합니다. Adobe AIR를 사용하여 개발한 iOS 응용 프로그램도 마찬가지입니다. 테스트 장치에 응용 프로그램을 설치하려면 Apple 개발 인증서를 사용하여 서명을 적용해야 합니다. 완성된 응용 프로그램을 배포하려면 배포 인증서를 사용하여 서명을 적용해야 합니다.

응용 프로그램에 서명하려면 ADT가 코드 서명 인증서와 관련 개인 키에 모두 액세스할 수 있어야 합니다. 인증서 파일 자체에는 개인 키가 포함되어 있지 않습니다. 인증서와 개인 키를 모두 포함하는 개인 정보 교환 파일(.p12 또는 .pfx)의 형태로 키 저장소를 만들어야 합니다. 177페이지의 "개발자 인증서를 P12 키 저장소 파일로 변환"을 참조하십시오.

인증서 서명 요청 생성

개발자 인증서를 받으려면 Apple iOS Provisioning Portal에서 제출하는 인증서 서명 요청을 생성합니다.

인증서 서명 요청 프로세스는 공개-개인 키 쌍을 생성합니다. 개인 키는 컴퓨터에 남습니다. 공개 키 및 식별 정보가 포함된 서명 요청을 Apple로 보냅니다. 이 경우 Apple이 인증 기관 역할을 합니다. Apple은 자체 World Wide Developer Relations 인증서를 사용하여 사용자의 인증서에 서명합니다.

Mac OS에서 인증서 서명 요청 생성

Mac OS에서 키체인 접근 응용 프로그램을 사용하여 코드 서명 요청을 생성할 수 있습니다. 키체인 접근 응용 프로그램은 [응용 프로그램] 디렉토리의 [유틸리티] 하위 디렉토리에 있습니다. 인증서 서명 요청을 생성하기 위한 지침은 Apple iOS Provisioning Portal에서 확인할 수 있습니다.

Windows에서 인증서 서명 요청 생성

Windows 개발자의 경우에도 Mac 컴퓨터에서 iPhone 개발자 인증서를 받는 것이 가장 쉽지만 Windows 컴퓨터에서 인증서를 받을 수도 있습니다. 먼저 OpenSSL을 사용하여 인증서 서명 요청(CSR 파일)을 만듭니다.

1 Windows 컴퓨터에 OpenSSL을 설치합니다. http://www.openssl.org/related/binaries.html로 이동합니다.

또한 Open SSL 다운로드 페이지에 나열된 Visual C++ 2008 재배포 가능 파일을 설치해야 할 수도 있습니다. 컴퓨터에 Visual C++를 설치할 필요는 없습니다.

2 Windows 명령 세션을 열고 CD 명령을 사용하여 OpenSSL bin 디렉토리(예: c:\OpenSSL\bin\)로 이동합니다.

3 명령줄에 다음을 입력하여 개인 키를 만듭니다.

```
openssl genrsa -out mykey.key 2048
```

이 개인 키 파일을 저장합니다. 이 파일은 나중에 사용합니다.

OpenSSL을 사용하는 경우 오류 메시지를 무시하지 마십시오. OpenSSL에서 오류 메시지를 생성하는 경우에도 파일을 출력할 수 있습니다. 그러나 해당 파일이 사용 가능한 상태가 아닐 수 있습니다. 오류가 표시되는 경우 구문을 확인하고 명령을 다시 실행합니다.

4 명령줄에 다음을 입력하여 CSR 파일을 만듭니다.

```
openssl req -new -key mykey.key -out CertificateSigningRequest.certSigningRequest  -subj
"/emailAddress=yourAddress@example.com, CN=John Doe, C=US"
```

전자 메일 주소, CN(인증서 이름) 및 C(국가) 값을 적절하게 바꿉니다.

5 iPhone 개발자 사이트에서 Apple에 CSR 파일을 업로드합니다. "iPhone 개발자 인증서 신청 및 프로비저닝 프로파일 만들기"를 참조하십시오.

개발자 인증서를 P12 키 저장소 파일로 변환

P12 키 저장소를 만들려면 Apple 개발자 인증서와 관련 개인 키를 단일 파일에 결합해야 합니다. 키 저장소 파일을 만드는 프로세스는 원래 인증서 서명 요청을 생성하는 데 사용하는 방법 그리고 개인 키가 저장된 위치에 따라 다릅니다.

Mac OS에서 iPhone 개발자 인증서를 P12 파일로 변환

Apple에서 Apple iPhone 인증서를 다운로드한 후에는 인증서를 P12 키 저장소 포맷으로 내보냅니다. Mac® OS에서 이 작업을 수행하는 방법은 다음과 같습니다.

1 키체인 접근 응용 프로그램(응용 프로그램/유틸리티 폴더에 있음)을 엽니다.

2 아직 키체인에 인증서를 추가하지 않은 경우 [파일] > [가져오기]를 선택합니다. 그런 다음 Apple에서 받은 인증서 파일(.cer 파일)로 이동합니다.

3 [키체인 접근]에서 [키] 카테고리를 선택합니다.

4 iPhone 개발 인증서에 연결된 개인 키를 선택합니다.

개인 키는 iPhone 개발자별로 식별됩니다. 즉, <이름> <성> 공용 인증서가 쌍을 이루고 있습니다.

5 iPhone 개발자 인증서를 Command 클릭하고 **"iPhone 개발자: 이름..."** 내보내기를 선택합니다.

6 개인 정보 교환(.p12) 파일 포맷으로 키 저장소를 저장합니다.

7 키 저장소를 사용하여 응용 프로그램에 서명하거나 이 키 저장소에 있는 키 및 인증서를 다른 키 저장소로 전송할 때 사용되는 암호를 만들라는 대화 상자가 나타납니다.

Windows에서 Apple 개발자 인증서를 P12 인증서로 변환

AIR for iOS 응용 프로그램을 개발하려면 P12 인증서 파일을 사용해야 합니다. 이 인증서는 Apple로부터 받은 Apple iPhone 개발자 인증서 파일을 기반으로 생성합니다.

1 Apple로부터 받은 개발자 인증서 파일을 PEM 인증서 파일로 변환합니다. OpenSSL bin 디렉토리에서 다음 명령줄 명령문을 실행합니다.

```
openssl x509 -in developer_identity.cer -inform DER -out developer_identity.pem -outform PEM
```

2 Mac 컴퓨터의 키체인에서 가져온 개인 키를 사용하는 경우 키를 PEM 키로 변환합니다.

```
openssl pkcs12 -nocerts -in mykey.p12 -out mykey.pem
```

3 이제 iPhone 개발자 인증서의 PEM 버전과 키를 사용하여 유효한 P12 파일을 생성할 수 있습니다.

```
openssl pkcs12 -export -inkey mykey.key -in developer_identity.pem -out iphone_dev.p12
```

Mac OS 키체인에서 가져온 키를 사용하는 경우 이전 단계에서 생성한 PEM 버전을 사용합니다. 그렇지 않으면 앞서 Windows에서 생성한 OpenSSL 키를 사용합니다.

ADT를 사용하여 서명되지 않은 AIR 중간 파일 만들기

-prepare 명령을 사용하여 서명되지 않은 AIR 중간 파일을 만듭니다. AIR 중간 파일은 ADT -sign 명령을 사용하여 서명되어야만 유효한 AIR 설치 파일이 만들어집니다.

-prepare 명령은 -package 명령과 동일한 플래그 및 매개변수를 사용합니다(서명 옵션은 예외). 유일한 차이점은 출력 파일이 서명되지 않는다는 점입니다. 생성되는 중간 파일의 파일 확장명은 airi입니다.

AIR 중간 파일에 서명하려면 ADT -sign 명령을 사용합니다. 155페이지의 “ADT prepare 명령”을 참조하십시오.

ADT -prepare 명령 예

```
adt -prepare unsignedMyApp.airi myApp.xml myApp.swf components.swc
```

ADT를 사용하여 AIR 중간 파일 서명

ADT를 사용하여 AIR 중간 파일에 서명하려면 -sign 명령을 사용합니다. 서명 명령은 AIR 중간 파일(확장명은 airi)에 대해서만 실행됩니다. AIR 파일에는 두 번 서명할 수 없습니다.

AIR 중간 파일을 만들려면 ADT -prepare 명령을 사용합니다. 155페이지의 “ADT prepare 명령”을 참조하십시오.

AIRI 파일 서명

❖ ADT -sign 명령을 다음과 같은 구문으로 사용합니다.

```
adt -sign SIGNING_OPTIONS airi_file air_file
```

SIGNING_OPTIONS 서명 옵션은 AIR 파일 서명에 사용할 개인 키 및 인증서를 식별합니다. 이 옵션에 대한 자세한 내용은161페이지의 “ADT 코드 서명 옵션”을 참조하십시오.

airi_file 서명할 서명되지 않은 AIR 중간 파일 경로입니다.

air_file 만들 AIR 파일의 이름입니다.

ADT -sign 명령 예

```
adt -sign -storetype pkcs12 -keystore cert.p12 unsignedMyApp.airi myApp.air
```

자세한 내용은 156페이지의 “ADT sign 명령”을 참조하십시오.

업데이트된 버전의 AIR 응용 프로그램에 서명

기존 AIR 응용 프로그램의 업데이트 버전을 만들 때마다 업데이트된 응용 프로그램에 서명합니다. 최상의 경우는 이전 버전에 서명할 때 사용했던 것과 동일한 인증서를 사용하여 업데이트된 버전에 서명하는 것입니다. 이 경우 서명은 처음 응용 프로그램에 한 서명과 정확하게 동일합니다.

응용 프로그램의 이전 버전에 서명할 때 사용한 인증서가 만료되어 갱신되거나 교체된 경우 갱신된 인증서 또는 신규(교체) 인증서를 사용하여 업데이트 버전에 서명할 수 있습니다. 이렇게 하려면 응용 프로그램에 새 인증서로 서명하고 **이와 함께** 원래 인증서를 사용하여 마이그레이션 서명을 적용합니다. 마이그레이션 서명은 원래 인증서 소유자가 업데이트를 제작했음을 검증합니다.

마이그레이션 서명을 적용하기 전에 다음 사항을 고려하십시오.

- 마이그레이션 서명을 적용하려면 원래 인증서가 아직 유효하거나, 만료된 후 365일이 지나지 않은 상태여야 합니다. 이 기간을 '유예 기간'이라고 하며, 향후 이 기간의 길이가 변경될 수 있습니다.

  참고: AIR 2.6까지는 유예 기간이 180일이었습니다.

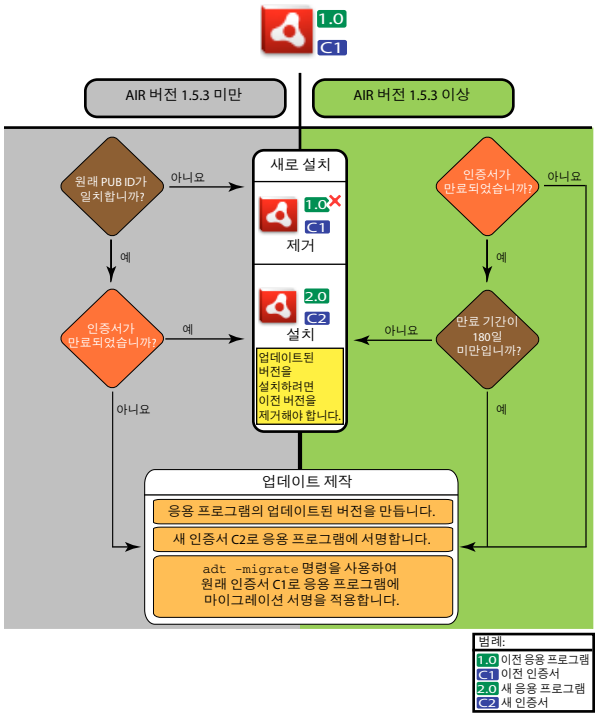
- 인증서가 만료되고 365일의 유예 기간이 경과된 후에는 마이그레이션 서명을 적용할 수 없습니다. 이 경우 사용자가 업데이트된 버전을 설치하기 전에 기존 버전을 제거해야 합니다.
- 365일의 유예 기간은 응용 프로그램 설명자 네임스페이스에 AIR 버전 1.5.3 이상이 지정된 응용 프로그램에만 적용됩니다.

중요: 만료된 인증서의 마이그레이션 서명을 사용하여 업데이트에 서명하는 것은 임시 방편입니다. 완벽하게 처리하려면 응용 프로그램 업데이트의 배포를 관리하기 위한 표준화된 서명 작업 과정을 만드십시오. 예를 들어 최신 인증서로 각 업데이트에 서명하고 이전 업데이트 서명에 사용한 인증서(있는 경우)로 마이그레이션 인증서를 적용합니다. 각 업데이트를 사용자가 응용 프로그램을 다운로드할 수 있도록 각각의 URL에 업로드합니다. 자세한 내용은 235페이지의 "응용 프로그램 업데이트를 위한 서명 작업 과정"을 참조하십시오.

다음 표와 그림은 마이그레이션 서명의 작업 과정을 정리한 것입니다.

시나리오	원래 인증서의 상태	개발자 작업	사용자 작업
Adobe AIR 런타임 버전 1.5.3 이상을 기반으로 하는 응용 프로그램	유효	최신 버전의 AIR 응용 프로그램을 제작합니다.	작업이 필요하지 않습니다. 응용 프로그램이 자동으로 업그레이드됩니다.
	만료되었지만 365일의 유예 기간이 끝나지 않음	새 인증서를 사용하여 응용 프로그램에 서명합니다. 만료된 인증서를 사용하여 마이그레이션 서명을 적용합니다.	작업이 필요하지 않습니다. 응용 프로그램이 자동으로 업그레이드됩니다.
	만료되었으며 유예 기간이 아님	마이그레이션 서명을 AIR 응용 프로그램 업데이트에 적용할 수 없습니다. 대신 새 인증서를 사용하여 다른 버전의 AIR 응용 프로그램을 제작해야 합니다. 사용자는 기존 버전의 AIR 응용 프로그램을 제거한 후에 새 버전을 설치할 수 있습니다.	현재 버전의 AIR 응용 프로그램을 제거하고 최신 버전을 설치합니다.

시나리오	원래 인증서의 상태	개발자 작업	사용자 작업
• Adobe AIR 런타임 버전 1.5.2 이하를 기반으로 하는 응용 프로그램 • 업데이트 응용 프로그램 설명자의 제작자 ID가 이전 버전의 제작자 ID와 일치합니다.	유효	최신 버전의 AIR 응용 프로그램을 제작합니다.	작업이 필요하지 않습니다. 응용 프로그램이 자동으로 업그레이드 됩니다.
	만료되었으며 유예 기간이 아님	마이그레이션 서명을 AIR 응용 프로그램 업데이트에 적용할 수 없습니다. 대신 새 인증서를 사용하여 다른 버전의 AIR 응용 프로그램을 제작해야 합니다. 사용자는 기존 버전의 AIR 응용 프로그램을 제거한 후에 새 버전을 설치할 수 있습니다.	현재 버전의 AIR 응용 프로그램을 제거하고 최신 버전을 설치합니다.
• Adobe AIR 런타임 버전 1.5.2 이하를 기반으로 하는 응용 프로그램 • 업데이트 응용 프로그램 설명자의 제작자 ID가 이전 버전의 제작자 ID와 **일치하지 않습니다**.	모두	유효한 인증서를 사용하여 AIR 응용 프로그램에 서명하고 AIR 응용 프로그램 최신 버전을 제작합니다.	현재 버전의 AIR 응용 프로그램을 제거하고 최신 버전을 설치합니다.



업데이트를 위한 서명 작업 과정

AIR 응용 프로그램을 마이그레이션하여 새 인증서 사용하기

응용 프로그램을 업데이트하면서 AIR 응용 프로그램을 새 인증서로 마이그레이션하려면 다음과 같이 합니다.

1 응용 프로그램의 업데이트를 만듭니다.

2 **새** 인증서를 사용하여 업데이트 AIR 파일을 패키지화하고 서명합니다.

3 -migrate 명령을 사용하여 **원래** 인증서로 AIR 파일을 다시 서명합니다.

-migrate 명령으로 서명된 AIR 파일은 이전 인증서로 서명된 이전 버전을 업데이트할 때 사용할 뿐 아니라 응용 프로그램 새 버전을 설치할 때도 사용할 수 있습니다.

참고: 1.5.3 미만의 AIR 버전용으로 제작된 응용 프로그램을 업데이트하는 경우 응용 프로그램 설명자에서 원래 제작자 ID를 지정하십시오. 그렇지 않으면 응용 프로그램 사용자가 이전 버전을 제거한 후 업데이트를 설치해야 합니다.

ADT -migrate 명령을 다음과 같은 구문으로 사용합니다.

```
adt -migrate SIGNING_OPTIONS air_file_in air_file_out
```

- **SIGNING_OPTIONS** 서명 옵션은 AIR 파일 서명에 사용할 개인 키 및 인증서를 식별합니다. 이 옵션으로 **원래** 서명 인증서를 식별해야 합니다. 이러한 옵션에 대한 자세한 내용은 161페이지의 “ADT 코드 서명 옵션”에서 설명합니다.
- **air_file_in 새** 인증서를 사용하여 서명한 업데이트의 AIR 파일입니다.
- **air_file_out** 만들 AIR 파일입니다.

참고: 입력 및 출력 AIR 파일에 사용되는 파일 이름은 서로 달라야 합니다.

다음 예제에서는 ADT를 -migrate 플래그와 함께 호출하여 마이그레이션 서명을 업데이트된 AIR 응용 프로그램 버전에 적용하는 것을 보여 줍니다.

```
adt -migrate -storetype pkcs12 -keystore cert.p12 myAppIn.air myApp.air
```

참고: -migrate 명령은 AIR 1.1 릴리스의 ADT에 추가되었습니다.

기본 설치 프로그램 AIR 응용 프로그램을 마이그레이션하여 새 인증서 사용하기

기본 설치 프로그램으로 제작된 AIR 응용 프로그램(예: 기본 확장 API를 사용하는 응용 프로그램)은 .air 파일이 아니라 플랫폼별 기본 응용 프로그램이기 때문에 ADT -migrate 명령을 사용하여 서명할 수 없습니다. 대신 기본 확장으로 제작된 AIR 응용 프로그램을 새 인증서로 마이그레이션하려면 다음과 같이 합니다.

1 응용 프로그램의 업데이트를 만듭니다.

2 응용 프로그램 설명자 파일(app.xml)에서 <supportedProfiles> 태그에 데스크톱 프로파일 및 extendedDesktop 프로파일이 모두 포함되어야 합니다. 그렇지 않으면 응용 프로그램 설명자에서 <supportedProfiles> 태그를 제거하십시오.

3 ADT -package 명령을 **새** 인증서와 함께 사용하여 업데이트된 응용 프로그램을 **.air 파일**로 패키지화하고 서명합니다.

4 ADT -migrate 명령을 **원래** 인증서와 함께 사용하여 마이그레이션 인증서를 .air 파일에 적용합니다(앞의 180페이지의 “AIR 응용 프로그램을 마이그레이션하여 새 인증서 사용하기” 단락 참조).

5 ADT -package 명령을 -target 기본 플래그와 함께 사용하여 .air 파일을 기본 설치 프로그램으로 패키지화합니다. 응용 프로그램이 이미 서명되어 있기 때문에 이 단계에서 서명 인증서를 지정하지 않습니다.

다음 예제에서는 이 프로세스의 3~5단계를 보여 줍니다. 예제의 코드는 -package 명령으로 ADT를 호출하고, -migrate 명령으로 ADT를 호출한 다음 다시 -package 명령으로 ADT를 호출하여 AIR 응용 프로그램의 업데이트된 버전을 기본 설치 프로그램으로 패키지화합니다.

```
adt -package -storetype pkcs12 -keystore new_cert.p12 myAppUpdated.air myApp.xml myApp.swf
adt -migrate -storetype pkcs12 -keystore original_cert.p12 myAppUpdated.air myAppMigrate.air
adt -package -target native myApp.exe myAppMigrate.air
```

기본 확장을 사용하는 AIR 응용 프로그램을 마이그레이션하여 새 인증서 사용하기

ADT -migrate 명령으로는 기본 확장을 사용하는 AIR 응용 프로그램에 서명할 수 없습니다. 또한 이 응용 프로그램은 중간 .air 파일로 제작할 수 없기 때문에 기본 설치 프로그램 AIR 응용 프로그램을 마이그레이션하는 절차로 마이그레이션할 수도 없습니다. 대신 기본 확장을 사용하는 AIR 응용 프로그램을 새 인증서로 마이그레이션하려면 다음과 같이 합니다.

1 응용 프로그램의 업데이트를 만듭니다.

2 ADT -package 명령을 사용하여 업데이트된 기본 설치 프로그램을 패키지화하고 서명합니다. **새** 인증서로 응용 프로그램을 패키지화하고 **원래** 인증서를 나타내는 -migrate 플래그를 포함합니다.

다음 구문을 사용하여 ADT -package 명령을 -migrate 플래그와 함께 호출합니다.

```
adt -package AIR_SIGNING_OPTIONS -migrate MIGRATION_SIGNING_OPTIONS -target package_type
NATIVE_SIGNING_OPTIONS output app_descriptor FILE_OPTIONS
```

- **AIR_SIGNING_OPTIONS** 서명 옵션은 AIR 파일 서명에 사용할 개인 키 및 인증서를 식별합니다. 이 옵션은 **새** 서명 인증서를 식별하며 자세한 내용은 161페이지의 “ADT 코드 서명 옵션”에서 설명합니다.
- **MIGRATION_SIGNING_OPTIONS** 서명 옵션은 AIR 파일 서명에 사용할 개인 키 및 인증서를 식별합니다. 이 옵션은 **원래** 서명 인증서를 식별하며 자세한 내용은 161페이지의 “ADT 코드 서명 옵션”에서 설명합니다.
- 그 외 나머지 옵션은 기본 설치 프로그램 AIR 응용 프로그램 패키지화에 사용되는 옵션과 동일하며 자세한 내용은 150페이지의 “ADT package 명령”에서 설명합니다.

다음 예제에서는 ADT를 -package 명령 및 -migrate 플래그와 함께 호출하여 기본 확장을 사용하는 AIR 응용 프로그램의 업데이트된 버전을 패키지화하고 마이그레이션 서명을 업데이트에 적용합니다.

```
adt -package -storetype pkcs12 -keystore new_cert.p12 -migrate -storetype pkcs12 -keystore original_cert.p12
-target native myApp.exe myApp.xml myApp.swf
```

참고: -package 명령의 -migrate 플래그는 AIR 3.6 이상의 ADT에서 사용할 수 있습니다.

ADT를 사용하여 자체 서명된 인증서 만들기

자체 서명된 인증서를 사용하여 유효한 AIR 설치 파일을 만들 수 있습니다. 하지만 자체 서명된 인증서는 사용자의 보안을 제한적으로만 보장하고, 자체 서명된 인증서의 신뢰성은 확인할 수 없습니다. 자체 서명된 AIR 파일을 설치하면 사용자에게 제작자 정보가 [알 수 없음]으로 표시됩니다. ADT에 의해 생성되는 인증서는 5년 동안 유효합니다.

자체 서명된 인증서로 서명한 AIR 응용 프로그램의 업데이트를 만들 경우 동일한 인증서를 사용하여 원래 및 업데이트 AIR 파일을 모두 서명해야 합니다. ADT에서 생성된 인증서는 동일한 매개변수를 사용한 경우에도 항상 고유합니다. 따라서 ADT에서 생성된 인증서로 업데이트를 자체 서명하려는 경우 원래 인증서를 안전한 위치에 보관하십시오. 또한 ADT에서 생성된 원래 인증서가 만료되면 업데이트된 AIR 파일을 만들 수 없습니다. 다른 인증서로 새 응용 프로그램을 제작할 수 있으나 동일한 응용 프로그램의 새 버전은 제작할 수 없습니다.

중요: 자체 서명된 인증서의 제한으로 인해 공개적으로 출시되는 AIR 응용 프로그램에 서명할 경우 신뢰할 수 있는 인증 기관의 상업용 인증서를 사용하는 것이 좋습니다.

ADT에서 생성된 인증서 및 연관된 개인 키는 PKCS12 유형의 키 저장소 파일에 저장됩니다. 지정된 암호는 키 저장소가 아닌 키 자체에 설정됩니다.

인증서 생성 예제

```
adt -certificate -cn SelfSign -ou QE -o "Example, Co" -c US 2048-RSA newcert.p12 39#wnetx3tl
adt -certificate -cn ADigitalID 1024-RSA SigningCert.p12 39#wnetx3tl
```

이 인증서를 사용하여 AIR 파일에 서명하려면 ADT -package 또는 -prepare 명령과 함께 다음과 같은 서명 옵션을 사용합니다.

```
-storetype pkcs12 -keystore newcert.p12 -storepass 39#wnetx3tl
-storetype pkcs12 -keystore SigningCert.p12 -storepass 39#wnetx3tl
```

참고: Java 버전 1.5 이상의 경우 PKCS12 인증서 파일을 보호하는 데 사용되는 암호에 상위 ASCII 문자를 사용할 수 없습니다. 암호에는 일반 ASCII 문자만 사용하십시오.

14장: AIR 응용 프로그램 설명자 파일

모든 AIR 응용 프로그램에는 응용 프로그램 설명자 파일이 필요합니다. 응용 프로그램 설명자 파일은 응용 프로그램의 기본 속성을 정의하는 XML 문서입니다.

AIR를 지원하는 대부분의 개발 환경은 프로젝트를 만들 때 자동으로 응용 프로그램 설명자를 생성합니다. 그렇지 않은 경우 직접 설명자 파일을 생성해야 합니다. 샘플 설명자 파일(descriptor-sample.xml)은 AIR 및 Flex SDK의 samples 디렉토리에 있습니다.

응용 프로그램 설명자 파일에는 모든 파일 이름을 사용할 수 있습니다. 응용 프로그램을 패키지화하면 응용 프로그램 설명자 파일이 application.xml로 이름이 변경되고 AIR 패키지의 특수 디렉토리 내에 배치됩니다.

응용 프로그램 설명자 예제

다음 응용 프로그램 설명자 문서에서는 대부분의 AIR 응용 프로그램에서 사용하는 기본 속성을 설정합니다.

```
<?xml version="1.0" encoding="utf-8" ?>
<application xmlns="http://ns.adobe.com/air/application/3.0">
    <id>example.HelloWorld</id>
    <versionNumber>1.0.1</versionNumber>
    <filename>Hello World</filename>
    <name>Example Co. AIR Hello World</name>
     <description>
        <text xml:lang="en">This is an example.</text>
        <text xml:lang="fr">C'est un exemple.</text>
        <text xml:lang="es">Esto es un ejemplo.</text>
    </description>
    <copyright>Copyright (c) 2010 Example Co.</copyright>
    <initialWindow>
        <title>Hello World</title>
        <content>
            HelloWorld.swf
        </content>
    </initialWindow>
    <icon>
        <image16x16>icons/smallIcon.png</image16x16>
        <image32x32>icons/mediumIcon.png</image32x32>
        <image48x48>icons/bigIcon.png</image48x48>
        <image128x128>icons/biggerIcon.png</image128x128>
    </icon>
</application>
```

응용 프로그램이 SWF 파일이 아닌 HTML 파일을 루트 내용으로 사용하는 경우 <content> 요소만 달라집니다.

```
<content>
    HelloWorld.html
</content>
```

응용 프로그램 설명자 변경 사항

다음 AIR 릴리스에서 AIR 응용 프로그램 설명자가 변경되었습니다.

AIR 1.1 설명자 변경 사항

응용 프로그램에서 text 요소를 사용하여 name 및 description 요소를 지역화하는 것이 허용되었습니다.

AIR 1.5 설명자 변경 사항

contentType이 fileType의 필수 자식이 되었습니다.

AIR 1.5.3 설명자 변경 사항

응용 프로그램이 제작자 ID 값을 지정할 수 있도록 publisherID 요소가 추가되었습니다.

AIR 2.0 설명자 변경 사항

추가됨:

- aspectRatio
- autoOrients
- fullScreen
- image29x29(imageNxN 참조)
- image57x57
- image72x72
- image512x512
- iPhone
- renderMode
- supportedProfiles

AIR 2.5 설명자 변경 사항

제거됨: version

추가됨:

- android
- extensionID
- extensions
- image36x36(imageNxN 참조)
- manifestAdditions
- versionLabel
- versionNumber

AIR 2.6 설명자 변경 사항

추가됨:

- image114x114(imageNxN 참조)
- requestedDisplayResolution
- softKeyboardBehavior

AIR 3.0 설명자 변경 사항

추가됨:

- colorDepth
- **direct**(renderMode의 유효한 값)
- renderMode는 더 이상 데스크톱 플랫폼에서 무시되지 않습니다.
- Android <uses-sdk> 요소를 지정할 수 있습니다. 이전에는 이 기능이 허용되지 않았습니다.

AIR 3.1 설명자 변경 사항

추가됨:

- 196페이지의 "Entitlements"

AIR 3.2 설명자 변경 사항

추가됨:

- depthAndStencil
- supportedLanguages

AIR 3.3 설명자 변경 사항

추가됨:

- aspectRatio에 이제 ANY 옵션이 포함됩니다.

AIR 3.4 설명자 변경 사항

추가됨:

- image50x50(204페이지의 "imageNxN" 참조)
- image58x58(204페이지의 "imageNxN" 참조)
- image100x100(204페이지의 "imageNxN" 참조)
- image1024x1024(204페이지의 "imageNxN" 참조)

AIR 3.6 설명자 변경 사항

추가됨:

- 이제 207페이지의 "iPhone" 요소의 214페이지의 "requestedDisplayResolution"에 excludeDevices 특성이 포함되므로 고해상도를 사용하는 iOS 대상 또는 표준 해상도를 사용하는 iOS 대상을 지정할 수 있습니다.
- 205페이지의 "initialWindow"의 새 214페이지의 "requestedDisplayResolution" 요소는 고해상도 디스플레이를 사용하는 Mac 등의 데스크톱 플랫폼에 고해상도를 사용할 것인지 또는 표준 해상도를 사용할 것인지 지정합니다.

AIR 3.7 설명자 변경 사항

추가됨:

- 207페이지의 “iPhone” 요소는 이제 런타임에 로드할 SWF 목록을 지정할 수 있도록 198페이지의 “externalSwfs” 요소를 제공합니다.
- 207페이지의 “iPhone” 요소는 이제 지정한 장치 집합에 CPU 렌더링 모드를 강제 적용할 수 있도록 201페이지의 “forceCPURenderModeForDevices” 요소를 제공합니다.

응용 프로그램 설명자 파일 구조

응용 프로그램 설명자 파일은 다음 구조를 갖춘 XML 문서입니다.

```
<application xmlns="http://ns.adobe.com/air/application/3.0">
    <allowBrowserInvocation>...<allowBrowserInvocation>
    <android>
        <colorDepth>...</colorDepth>
        <manifestAdditions
                <manifest>...</manifest>
            ]]>
        </manifestAdditions
    </android>
    <copyright>...</copyright>
    customUpdateUI>...</
    <description>
        <text xml:lang="...">...</text>
    </description>
    <extensions>
        <extensionID>...</extensionID>
    </extensions>
    <filename>...</filename>
    <fileTypes>
        <fileType>
            <contentType>...</contentType>
            <description>...</description>
            <extension>...</extension>
            <icon>
                <imageNxN>...</imageNxN>
            </icon>
            <name>...</name>
        </fileType>
    </fileTypes>
    <icon>
        <imageNxN>...</imageNxN>
    </icon>
    <id>...</id>
    <initialWindow>
        <aspectRatio>...</aspectRatio>
        <autoOrients>...</autoOrients>
        <content>...</content>
        <depthAndStencil>...</depthAndStencil>
        <fullScreen>...</fullScreen>
        <height>...</height>
        <maximizable>...</maximizable>
        <maxSize>...</maxSize>
        <minimizable>...</minimizable>
        <minSize>...</minSize>
        <renderMode>...</renderMode>
        <requestedDisplayResolution>...</requestedDisplayResolution>
```

```
        <resizable>...</resizable>
        <softKeyboardBehavior>...</softKeyboardBehavior>
        <systemChrome>...</systemChrome>
        <title>...</title>
        <transparent>...</transparent>
        <visible>...</visible>
        <width>...</width>
        <x>...</x>
        <y>...</y>
    </initialWindow>
    <installFolder>...</installFolder>
    <iPhone>
        <Entitlements>...</Entitlements>
        <InfoAdditions>...</InfoAdditions>
        <requestedDisplayResolution>...</requestedDisplayResolution>
        <forceCPURenderModeForDevices>...</forceCPURenderModeForDevices>
        <externalSwfs>...</externalSwfs>
    </iPhone>
    <name>
        <text xml:lang="...">...</text>
    </name>
    <programMenuFolder>...</programMenuFolder>
    <publisherID>...</publisherID>
    <215 페이지의 "supportedLanguages">...</215 페이지의 "supportedLanguages">
    <supportedProfiles>...</supportedProfiles>
    <versionNumber>...</versionNumber>
    <versionLabel>...</versionLabel>
</application>
```

AIR 응용 프로그램 설명자 요소

다음의 요소 사전에서는 AIR 응용 프로그램 설명자 파일의 각 유효 요소에 대해 설명합니다.

allowBrowserInvocation

Adobe AIR 1.0 이상 - 선택 사항

AIR 인 브라우저 API가 응용 프로그램을 감지하고 실행할 수 있도록 합니다.

이 값을 true로 설정하는 경우 보안 영향을 고려해야 합니다. 이러한 보안 영향은 브라우저에서 AIR 응용 프로그램 호출 (ActionScript 개발자용) 및 Invoking an AIR application from the browser(HTML 개발자용)에 설명되어 있습니다.

자세한 내용은 232페이지의 "브라우저에서 설치된 AIR 응용 프로그램 시작"을 참조하십시오.

부모 요소:188페이지의 "application"

자식 요소: 없음

내용

true 또는 false(기본값)

예제

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

android

Adobe AIR 2.5 이상 - 선택 사항

Android 매니페스트 파일에 요소를 추가할 수 있도록 합니다. AIR는 모든 APK 패키지에 대해 AndroidManifest.xml 파일을 만듭니다. AIR 응용 프로그램 설명자에서 android 요소를 사용하여 항목을 더 추가할 수 있습니다. Android를 제외한 모든 플랫폼에서 무시됩니다.

부모 요소:188페이지의 “application”

자식 요소:

- 192페이지의 “colorDepth”
- 209페이지의 “manifestAdditions”

내용

Android 응용 프로그램 매니페스트에 추가할 Android 관련 속성을 정의하는 요소

예제

```
<android>
    <manifestAdditions>
        ...
    </manifestAdditions>
</android>
```

기타 도움말 항목

69페이지의 “Android 설정”

AndroidManifest.xml 파일

application

Adobe AIR 1.0 이상 - 필수 사항

AIR 응용 프로그램 설명자 문서의 루트 요소입니다.

부모 요소: 없음

자식 요소:

- 187페이지의 “allowBrowserInvocation”
- 188페이지의 “android”
- 193페이지의 “copyright”
- 194페이지의 “customUpdateUI”
- 195페이지의 “description”
- 197페이지의 “extensions”
- 198페이지의 “filename”
- 200페이지의 “fileTypes”
- 203페이지의 “icon”
- 203페이지의 “id”
- 205페이지의 “initialWindow”

- 207페이지의 “installFolder”
- 207페이지의 “iPhone”
- 211페이지의 “name”
- 212페이지의 “programMenuFolder”
- 213페이지의 “publisherID”
- 215페이지의 “supportedLanguages”
- 216페이지의 “supportedProfiles”
- 218페이지의 “version”
- 219페이지의 “versionLabel”
- 219페이지의 “versionNumber”

특성

minimumPatchLevel - 이 응용 프로그램에 필요한 AIR 런타임 최소 패치 수준입니다.

xmlns - 이 XML 네임스페이스 특성에 따라 응용 프로그램의 필수 AIR 런타임 버전이 결정됩니다.

이 네임스페이스는 보조 패치가 아니라 AIR의 주요 릴리스가 출시될 때마다 변경됩니다. "3.0"과 같은 네임스페이스의 마지막 세그먼트는 응용 프로그램에 필요한 런타임 버전을 나타냅니다.

주요 AIR 릴리스의 xmlns 값은 다음과 같습니다.

```
xmlns="http://ns.adobe.com/air/application/1.0"
xmlns="http://ns.adobe.com/air/application/1.1"
xmlns="http://ns.adobe.com/air/application/1.5"
xmlns="http://ns.adobe.com/air/application/1.5.2"
xmlns="http://ns.adobe.com/air/application/1.5.3"
xmlns="http://ns.adobe.com/air/application/2.0"
xmlns="http://ns.adobe.com/air/application/2.5"
xmlns="http://ns.adobe.com/air/application/2.6"
xmlns="http://ns.adobe.com/air/application/2.7"
xmlns="http://ns.adobe.com/air/application/3.0"
xmlns="http://ns.adobe.com/air/application/3.1"
xmlns="http://ns.adobe.com/air/application/3.2"
xmlns="http://ns.adobe.com/air/application/3,3"
xmlns="http://ns.adobe.com/air/application/3.4"
xmlns="http://ns.adobe.com/air/application/3.5"
xmlns="http://ns.adobe.com/air/application/3.6"
xmlns="http://ns.adobe.com/air/application/3.7"
```

SWF 기반 응용 프로그램의 경우 응용 프로그램 설명자에 지정된 AIR 런타임 버전에 따라 응용 프로그램의 초기 내용으로 로드할 수 있는 최대 SWF 버전이 결정됩니다. AIR 1.0 또는 AIR 1.1이 지정된 응용 프로그램에서는 AIR 2 런타임을 사용하여 실행하는 경우에도 초기 내용으로 SWF9(Flash Player 9) 파일만 사용할 수 있습니다. AIR 1.5(또는 그 이상)가 지정된 응용 프로그램에서는 초기 내용으로 SWF9 또는 SWF10(Flash Player 10) 파일을 사용할 수 있습니다.

SWF 버전에 따라 사용할 수 있는 AIR 및 Flash Player API 버전이 달라집니다. SWF9 파일이 AIR 1.5 응용 프로그램의 초기 내용으로 사용되는 경우 응용 프로그램에서 AIR 1.1 및 Flash Player 9 API에만 액세스할 수 있습니다. 이 경우 AIR 2.0 또는 Flash Player 10.1에서 기존 API에 이루어진 비헤이비어 변경 사항도 적용되지 않습니다. 단, API에 대한 중요한 보안 관련 변경 사항은 이 원칙에 대한 예외로서 현재 또는 향후 런타임 패치에서 소급 적용할 수 있습니다.

HTML 기반 응용 프로그램의 경우 응용 프로그램 설명자에 지정된 런타임 버전으로 응용 프로그램에 사용할 수 있는 AIR 및 Flash Player API 버전이 결정됩니다. HTML, CSS 및 JavaScript 비헤이비어는 응용 프로그램 설명자에 의해 결정되는 것이 아니라 항상 설치된 AIR 런타임에 사용된 Webkit 버전에 의해 결정됩니다.

AIR 응용 프로그램에서 SWF 내용을 로드할 때 해당 내용에 사용할 수 있는 AIR 및 Flash Player API 버전은 내용을 로드하는 방법에 따라 달라집니다. 유효한 버전은 응용 프로그램 설명자 네임스페이스로 결정되거나, 로딩하는 내용의 버전에 따라 결정되거나, 로드된 내용의 버전에 따라 결정됩니다. 다음 표에서는 내용을 로드하는 방법에 따라 API 버전이 어떻게 결정되는지 보여 줍니다.

내용 로드 방법	**API 버전 결정 방법**
초기 내용, SWF 기반 응용 프로그램	**로드되는** SWF 파일 버전
초기 내용, HTML 기반 응용 프로그램	응용 프로그램 설명자 네임스페이스
SWF 내용에서 로드되는 SWF	**로드하는** 내용의 버전
<script> 태그를 사용하여 HTML 내용에서 로드되는 SWF 라이브러리	응용 프로그램 설명자 네임스페이스
AIR 또는 Flash Player API(예: flash.display.Loader)를 사용하여 HTML에서 로드되는 SWF	응용 프로그램 설명자 네임스페이스
<object> 또는 <embed> 태그(또는 이에 해당하는 JavaScript API)를 사용하여 HTML 내용에서 로드되는 SWF	**로드되는** SWF 파일 버전

로드하는 내용과 다른 버전의 SWF 파일을 로드할 경우 두 가지 문제가 발생할 수 있습니다.

- 이전 버전의 SWF를 통해 최신 버전의 SWF 로드 - 로드된 내용에 있는 최신 버전의 AIR 및 Flash Player에 추가된 API에 대한 참조가 확인되지 않습니다.
- 최신 버전의 SWF를 통해 이전 버전의 SWF 로드 - 최신 버전의 AIR 및 Flash Player에서 변경된 API가 로드된 내용에 불필요한 방식으로 작동할 수 있습니다.

내용

application 요소에는 AIR 응용 프로그램의 속성을 정의하는 자식 요소가 포함되어 있습니다.

예제

```
<?xml version="1.0" encoding="utf-8" ?>
<application xmlns="http://ns.adobe.com/air/application/3.0">
    <id>HelloWorld</id>
    <version>2.0</version>
    <filename>Hello World</filename>
    <name>Example Co. AIR Hello World</name>
     <description>
        <text xml:lang="en">This is an example.</text>
        <text xml:lang="fr">C'est un exemple.</text>
        <text xml:lang="es">Esto es un ejemplo.</text>
    </description>
    <copyright>Copyright (c) 2010 Example Co.</copyright>
    <initialWindow>
        <title>Hello World</title>
        <content>
            HelloWorld.swf
        </content>
        <systemChrome>none</systemChrome>
        <transparent>true</transparent>
        <visible>true</visible>
        <minSize>320 240</minSize>
    </initialWindow>
    <installFolder>Example Co/Hello World</installFolder>
    <programMenuFolder>Example Co</programMenuFolder>
```

```
    <icon>
        <image16x16>icons/smallIcon.png</image16x16>
        <image32x32>icons/mediumIcon.png</image32x32>
        <image48x48>icons/bigIcon.png</image48x48>
        <image128x128>icons/biggestIcon.png</image128x128>
    </icon>
    <customUpdateUI>true</customUpdateUI>
    <allowBrowserInvocation>false</allowBrowserInvocation>
    <fileTypes>
        <fileType>
            <name>adobe.VideoFile</name>
            <extension>avf</extension>
            <description>Adobe Video File</description>
            <contentType>application/vnd.adobe.video-file</contentType>
            <icon>
                <image16x16>icons/avfIcon_16.png</image16x16>
                <image32x32>icons/avfIcon_32.png</image32x32>
                <image48x48>icons/avfIcon_48.png</image48x48>
                <image128x128>icons/avfIcon_128.png</image128x128>
            </icon>
        </fileType>
    </fileTypes>
</application>
```

aspectRatio

Adobe AIR 2.0 이상, iOS 및 Android - 선택 사항

응용 프로그램의 종횡비를 지정합니다.

지정하지 않으면 응용 프로그램이 장치의 "기본" 종횡비 및 방향으로 열립니다. 기본 방향은 장치마다 다릅니다. 일반적으로 휴대폰과 같은 화면이 작은 장치에서는 세로 종횡비입니다. iPad 타블렛 같은 일부 장치에서는 응용 프로그램이 현재 방향으로 열립니다. AIR 3.3 이상에서는 초기 표시뿐 아니라 전체 응용 프로그램에 이 점이 적용됩니다.

부모 요소:205페이지의 "initialWindow"

자식 요소: 없음

내용

portrait, landscape 또는 any

예제

```
<aspectRatio>landscape</aspectRatio>
```

autoOrients

Adobe AIR 2.0 이상, iOS 및 Android - 선택 사항

장치 자체의 실제 방향이 변경되는 경우 응용 프로그램 내 내용의 방향이 자동으로 조정되도록 할지 여부를 지정합니다. 자세한 내용은 스테이지 방향을 참조하십시오.

자동 방향을 사용하는 경우에는 Stage의 align 및 scaleMode 속성을 다음과 같이 설정하는 것이 좋습니다.

```
stage.align = StageAlign.TOP_LEFT;
stage.scaleMode = StageScaleMode.NO_SCALE;
```

이러한 설정은 응용 프로그램이 왼쪽 위 모서리를 중심으로 회전하도록 하고 응용 프로그램 내용의 크기가 자동으로 조절되는 것을 방지합니다. 다른 크기 조절 모드는 회전한 스테이지 크기에 맞게 내용을 조정하기는 하지만 해당 내용을 잘라내거나, 왜곡하거나, 과도하게 축소할 수 있습니다. 거의 언제나 내용을 직접 다시 그리거나 배치함으로써 더 나은 결과를 얻을 수 있습니다.

부모 요소:205페이지의 "initialWindow"

자식 요소: 없음

내용
true 또는 false(기본값)

예제
```
<autoOrients>true</autoOrients>
```

colorDepth

Adobe AIR 3 이상 - 선택 사항

16비트 색상을 사용할지 아니면 32비트 색상을 사용할지를 지정합니다.

16비트 색상을 사용하면 렌더링 성능이 증가하지만 색상 충실도가 떨어질 수 있습니다. AIR 3 이전에는 Android에 항상 16비트 색상이 사용되지만, AIR 3에서는 기본적으로 32비트 색상이 사용됩니다.

참고: 응용 프로그램에서 StageVideo 클래스를 사용하는 경우 32비트 색상을 사용해야 합니다.

부모 요소:188페이지의 "android"

자식 요소: 없음

내용
다음 값 중 하나:

- 16비트
- 32비트

예제
```
<android>
    <colorDepth>16bit</colorDepth>
    <manifestAdditions>...</manifestAdditions>
</android>
```

containsVideo

응용 프로그램에 비디오 내용이 포함될지 여부를 지정합니다.

부모 요소:188페이지의 "android"

자식 요소: 없음

내용
다음 값 중 하나:

- true
- false

예제

```
<android>
    <containsVideo>true</containsVideo>
    <manifestAdditions>...</manifestAdditions>
</android>
```

content

Adobe AIR 1.0 이상 - 필수 사항

content 요소에 대해 지정되는 값은 응용 프로그램의 기본 내용 파일에 대한 URL입니다. 이는 SWF 파일 또는 HTML 파일일 수 있습니다. URL은 응용 프로그램 설치 폴더의 루트를 기준으로 지정됩니다. ADL을 사용하여 AIR 응용 프로그램을 실행하는 경우 URL은 응용 프로그램 설명자 파일이 포함된 폴더를 기준으로 합니다. ADL의 root-dir 매개 변수를 사용하여 다른 루트 디렉토리를 지정할 수 있습니다.

부모 요소:205페이지의 “initialWindow”

자식 요소: 없음

내용

응용 프로그램 디렉토리에 상대적인 URL. content 요소의 값이 URL로 처리되기 때문에 내용 파일의 이름에 사용된 문자는 RFC 1738에 정의된 규칙에 따라 URL로 인코딩해야 합니다. 예를 들어 공백 문자는 %20으로 인코딩해야 합니다.

예제

```
<content>TravelPlanner.swf</content>
```

contentType

Adobe AIR 1.0부터 1.1까지 - 선택 사항, AIR 1.5 이상 - 필수 사항

contentType은 AIR 1.5부터 필수입니다(AIR 1.0 및 1.1에서는 선택 사항이었음). 이 속성은 일부 운영 체제에서 파일을 여는 데 가장 적합한 응용 프로그램을 손쉽게 찾을 수 있도록 합니다. 값은 파일 내용의 MIME 유형이어야 합니다. 해당 파일 형식이 이미 등록되어 있고 지정된 MIME 유형을 갖는 경우 Linux에서 값이 무시됩니다.

부모 요소:199페이지의 “fileType”

자식 요소: 없음

내용

MIME 유형 및 하위 유형. MIME 유형에 대한 자세한 내용은 RFC2045를 참조하십시오.

예제

```
<contentType>text/plain</contentType>
```

copyright

Adobe AIR 1.0 이상 - 선택 사항

AIR 응용 프로그램에 대한 저작권 정보입니다. Mac OS에서 저작권 텍스트는 설치된 응용 프로그램의 [정보] 대화 상자에 나타납니다. Mac OS에서는 저작권 정보가 응용 프로그램에 대한 Info.plist 파일의 NSHumanReadableCopyright 필드에도 사용됩니다.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용
응용 프로그램 저작권 정보를 포함하는 문자열

예제
```
<copyright>© 2010, Examples, Inc.All rights reserved.</copyright>
```

customUpdateUI

Adobe AIR 1.0 이상 - 선택 사항

응용 프로그램에서 자체적인 업데이트 대화 상자를 제공할지 여부를 나타냅니다. false인 경우 AIR는 사용자에게 표준 업데이트 대화 상자를 표시합니다. AIR 파일로 배포된 응용 프로그램만 내장된 AIR 업데이트 시스템을 사용할 수 있습니다.

설치된 응용 프로그램 버전의 customUpdateUI 요소가 true로 설정된 상태에서 사용자가 새 버전에 대한 AIR 파일을 두 번 클릭하거나 연속 설치 기능을 사용하여 응용 프로그램의 업데이트를 설치하면 런타임은 설치된 버전의 응용 프로그램을 열며, 기본 AIR 설치 프로그램은 열지 않습니다. 그러면 응용 프로그램 논리에서 업데이트 작업의 진행 방법을 결정할 수 있습니다. AIR 파일의 응용 프로그램 ID 및 제작자 ID가 설치된 응용 프로그램의 해당 값과 일치해야 업그레이드를 진행할 수 있습니다.

참고: 응용 프로그램이 이미 설치되어 있으며 사용자가 업데이트가 포함된 AIR 설치 파일을 두 번 클릭하거나 연속 설치 기능을 사용하여 응용 프로그램의 업데이트를 설치하는 경우에만 customUpdateUI 메커니즘이 작동합니다. customUpdateUI가 true인지 여부에 관계없이 필요에 따라 사용자 정의 UI를 표시하며 고유한 응용 프로그램 논리를 통해 업데이트를 다운로드 및 시작할 수 있습니다.

자세한 내용은 234페이지의 “AIR 응용 프로그램 업데이트”를 참조하십시오.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용
true 또는 false(기본값)

예제
```
<customUpdateUI>true</customUpdateUI>
```

depthAndStencil

Adobe AIR 3.2 이상 - 선택 사항

응용 프로그램에서 심도 또는 스텐실 버퍼를 사용해야 한다는 것을 나타냅니다. 일반적으로 이러한 버퍼는 3D 내용으로 작업하는 경우에 사용합니다. 기본적으로 이 요소의 값은 심도 및 스텐실 버퍼를 비활성화하는 false입니다. 이 요소가 필요한 이유는 응용 프로그램 시작 시 내용이 로드되기 전에 버퍼를 할당해야 하기 때문입니다.

이 요소의 설정은 enableDepthAndStencil 인수에 대해 Context3D.configureBackBuffer() 메서드로 전달되는 값과 일치해야 합니다. 값이 일치하지 않으면 AIR에서 오류가 발생합니다.

이 요소는 renderMode가 direct인 경우에만 적용할 수 있습니다. renderMode가 direct가 아닌 경우에는 ADT에서 오류 118이 발생합니다.

```
<depthAndStencil> element unexpected for render mode cpu.  It requires "direct" render mode.
```

부모 요소:205페이지의 “initialWindow”

자식 요소: 없음

내용
true 또는 false(기본값)

예제
```
<depthAndStencil>true</depthAndStencil>
```

description

Adobe AIR 1.0 이상 - 선택 사항

AIR 응용 프로그램 설치 프로그램에 표시되는 응용 프로그램에 대한 설명입니다.

다중 text 요소 대신 단일 텍스트 노드를 지정하는 경우 AIR 응용 프로그램 설치 프로그램은 시스템 언어와 상관없이 이 설명을 사용합니다. 그렇지 않은 경우 AIR 응용 프로그램 설치 프로그램은 사용되는 운영 체제의 사용자 인터페이스 언어에 가장 가까운 설명을 사용합니다. 예를 들어 응용 프로그램 설명자 파일의 description 요소에 en(영어) 로캘의 값이 포함된 설치를 살펴보면 사용자 시스템이 en(영어)을 사용자 인터페이스 언어로 식별하는 경우 AIR 응용 프로그램 설치 프로그램은 en 설명을 사용합니다. 시스템 사용자 인터페이스 언어가 en-US(미국 영어)인 경우에도 en 설명이 사용됩니다. 그러나 시스템 사용자 인터페이스 언어가 en-US이며 응용 프로그램 설명자 파일이 en-US 이름과 en-GB 이름을 모두 정의하는 경우 AIR 응용 프로그램 설치 프로그램은 en-US 값을 사용합니다. 응용 프로그램이 시스템 사용자 인터페이스 언어와 일치하는 설명을 정의하지 않는 경우 AIR 응용 프로그램 설치 프로그램은 응용 프로그램 설명자 파일에 정의된 첫 번째 description 값을 사용합니다.

다중 언어 응용 프로그램을 개발하는 방법에 대한 자세한 내용은 267페이지의 “AIR 응용 프로그램 지역화”를 참조하십시오.

부모 요소:188페이지의 “application”

자식 요소:217페이지의 “text”

내용
AIR 1.0 응용 프로그램 설명자 스키마를 사용하면 이름에 대해 여러 text 요소가 아닌 하나의 간단한 텍스트 노드만 정의할 수 있습니다.

AIR 1.1 이상에서는 description 요소에 다중 언어를 지정할 수 있습니다. 각 text 요소에 대한 xml:lang 특성은 RFC4646(http://www.ietf.org/rfc/rfc4646.txt)에 정의된 언어 코드를 지정합니다.

예제
간단한 텍스트 노드가 있는 설명:

```
<description>This is a sample AIR application.</description>
```

영어, 프랑스어 및 스페인어에 대한 지역화된 text 요소가 있는 설명(AIR 1.1 이상에서 유효):

```
<description>
    <text xml:lang="en">This is an example.</text>
    <text xml:lang="fr">C'est un exemple.</text>
    <text xml:lang="es">Esto es un ejemplo.</text>
</description>
```

description

Adobe AIR 1.0 이상 - 필수 사항

파일 유형 설명은 운영 체제를 통해 사용자에게 표시됩니다. 파일 유형 설명은 지역화할 수 없습니다.

참조: application 요소의 자식으로 사용될 때의 195페이지의 “description”

부모 요소:199페이지의 “fileType”

자식 요소: 없음

내용
파일 내용을 설명하는 문자열

예제
```
<description>PNG image</description>
```

embedFonts

AIR 응용 프로그램에서 StageText에 사용자 정의 글꼴을 사용할 수 있습니다. 이 요소는 선택 사항입니다.

부모 요소:188페이지의 “application”

자식 요소:200페이지의 “글꼴”

내용
embedFonts 요소는 font 요소를 개수에 관계없이 포함할 수 있습니다.

예제
```
<embedFonts>
   <font>
      <fontPath>ttf/space age.ttf</fontPath>
      <fontName>space age</fontName>
   </font>
   <font>
      <fontPath>ttf/xminus.ttf</fontPath>
      <fontName>xminus</fontName>
   </font>
</embedFonts>
```

Entitlements

Adobe AIR 3.1 이상 - iOS 전용, 선택 사항

iOS에서는 entitlements라는 속성을 사용하여 응용 프로그램이 추가 리소스 및 기능에 액세스하도록 지원합니다. Entitlements 요소를 사용하면 모바일 iOS 응용 프로그램에서 이 정보를 지정할 수 있습니다.

부모 요소:207페이지의 “iPhone”

자식 요소: iOS Entitlements.plist 요소

내용
응용 프로그램의 Entitlements.plist 설정으로 사용할 키-값 쌍을 지정하는 자식 요소가 포함됩니다. Entitlements 요소의 내용은 CDATA 블록으로 둘러싸여야 합니다. 자세한 내용은 iOS 개발자 라이브러리에서 Entitlement 키 참조를 검토하십시오.

예제

```
<iphone>
...
   <Entitlements>
      <![CDATA[
         <key>aps-environment</key>
         <string>development</string>
      ]]>
   </Entitlements>
</iphone>
```

extension

Adobe AIR 1.0 이상 - 필수 사항

파일 유형의 확장자 문자열입니다.

부모 요소:199페이지의 “fileType”

자식 요소: 없음

내용

파일 확장자 문자를 식별하는 문자열(점, “.” 제외)

예제

```
<extension>png</extension>
```

extensionID

Adobe AIR 2.5 이상

응용 프로그램에서 사용하는 ActionScript 확장의 ID를 지정합니다. ID는 확장 설명자 문서에 정의되어 있습니다.

부모 요소:197페이지의 “extensions”

자식 요소: 없음

내용

ActionScript 확장 ID를 식별하는 문자열

예제

```
<extensionID>com.example.extendedFeature</extensionID>
```

extensions

Adobe AIR 2.5 이상 - 선택 사항

응용 프로그램에서 사용하는 ActionScript 확장을 식별합니다.

부모 요소:188페이지의 “application”

자식 요소:197페이지의 “extensionID”

내용
확장 설명자 파일에 있는 ActionScript 확장 ID를 포함하는 자식 extensionID 요소

예제
```
<extensions>
    <extensionID>extension.first</extensionID>
    <extensionID>extension.next</extensionID>
    <extensionID>extension.last</extensionID>
</extensions>
```

externalSwfs

Adobe AIR 3.7 이상, iOS만 해당 - 선택 사항

원격 호스팅을 위해 ADT에 구성할 SWF 목록이 포함된 텍스트 파일 이름을 지정합니다. 응용 프로그램에 사용되는 하위 집합의 SWF를 패키지화하고 Loader.load() 메서드를 사용하여 나머지 외부(에셋 전용) SWF를 런타임에 로드하여 초기 응용 프로그램 다운로드 크기를 최소화할 수 있습니다. 이 기능을 사용하려면 ADT가 외부 로드된 SWF 파일에서 기본 응용 프로그램 SWF로 모든 ActionScript ByteCode(ABC)를 이동할 수 있도록 응용 프로그램을 패키지화해야 합니다. 그래야 SWF에는 에셋만 포함됩니다. 이는 응용 프로그램을 설치한 후 어떠한 코드 다운로드도 금지하는 Apple Store의 규칙을 준수합니다.

자세한 내용은 78페이지의 "외부 에셋 전용 SWF를 로드하여 다운로드 크기 최소화"를 참조하십시오.

부모 요소: 207페이지의 "iPhone", 205페이지의 "initialWindow"

자식 요소: 없음

내용
원격으로 호스팅되는 SWF 목록(행으로 구분됨)을 포함하는 텍스트 파일 이름입니다.

특성:
없음

예제
iOS:

```
<iPhone>
    <externalSwfs>FileContainingListofSWFs.txt</externalSwfs>
</iPhone>
```

filename

Adobe AIR 1.0 이상 - 필수 사항

응용 프로그램을 설치할 때 응용 프로그램의 파일 이름(확장명 제외)으로 사용할 문자열입니다. 이 응용 프로그램 파일은 런타임에 AIR 응용 프로그램을 시작합니다. name 값을 제공하지 않으면 filename이 설치 폴더의 이름으로도 사용됩니다.

부모 요소:188페이지의 "application"

자식 요소: 없음

내용
filename 속성에는 다양한 파일 시스템에서 파일 이름으로 사용할 수 없는 다음 문자를 제외한 모든 유니코드(UTF-8) 문자를 포함할 수 있습니다.

문자	16진수 코드
다양함	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

filename 값은 마침표로 끝낼 수 없습니다.

예제

```
<filename>MyApplication</filename>
```

fileType

Adobe AIR 1.0 이상 - 선택 사항

응용 프로그램이 등록할 수 있는 단일 파일 유형을 설명합니다.

부모 요소:200페이지의 “fileTypes”

자식 요소:

- 193페이지의 “contentType”
- 195페이지의 “description”
- 197페이지의 “extension”
- 203페이지의 “icon”
- 212페이지의 “name”

내용

파일 유형을 설명하는 요소

예제

```
<fileType>
    <name>foo.example</name>
    <extension>foo</extension>
    <description>Example file type</description>
    <contentType>text/plain</contentType>
    <icon>
        <image16x16>icons/fooIcon16.png</image16x16>
        <image48x48>icons/fooIcon48.png</imge48x48>
    <icon>
</fileType>
```

fileTypes

Adobe AIR 1.0 이상 - 선택 사항

fileTypes 요소를 사용하면 AIR 응용 프로그램을 연결할 수 있는 파일 유형을 선언할 수 있습니다.

AIR 응용 프로그램이 설치되면 선언된 모든 파일 유형이 운영 체제에 등록됩니다. 이러한 파일 유형이 아직 다른 응용 프로그램과 연결되어 있지 않은 경우 AIR 응용 프로그램과 연결됩니다. 파일 유형과 다른 응용 프로그램 간의 기존 연결을 재정의하려면 런타임에 NativeApplication.setAsDefaultApplication() 메서드를 사용합니다. 이때 사용자의 허락을 받는 것이 좋습니다.

참고: 런타임 메서드는 응용 프로그램 설명자에 선언된 파일 유형에 대한 연결만 관리할 수 있습니다.

fileTypes 요소는 선택 사항입니다.

부모 요소:188페이지의 “application”

자식 요소:199페이지의 “fileType”

내용

개수에 관계없이 fileType 요소를 포함할 수 있는 fileTypes 요소

예제

```
<fileTypes>
    <fileType>
        <name>adobe.VideoFile</name>
        <extension>avf</extension>
        <description>Adobe Video File</description>
        <contentType>application/vnd.adobe.video-file</contentType>
        <icon>
            <image16x16>icons/AIRApp_16.png</image16x16>
            <image32x32>icons/AIRApp_32.png</image32x32>
            <image48x48>icons/AIRApp_48.png</image48x48>
            <image128x128>icons/AIRApp_128.png</image128x128>
        </icon>
    </fileType>
</fileTypes>
```

글꼴

AIR 응용 프로그램에서 사용할 수 있는 단일 사용자 정의 글꼴에 대해 설명합니다.

부모 요소:196페이지의 “embedFonts”

자식 요소:200페이지의 “fontName”, 201페이지의 “fontPath”

내용

사용자 정의 글꼴 이름 및 해당 경로를 지정하는 요소입니다.

예제

```
<font>
   <fontPath>ttf/space age.ttf</fontPath>
   <fontName>space age</fontName>
</font>
```

fontName

사용자 정의 글꼴 이름을 지정합니다.

부모 요소:200페이지의 “글꼴”

자식 요소: 없음

내용
StageText.fontFamily에 지정할 사용자 정의 글꼴의 이름입니다.

예제
```
<fontName>space age</fontName>
```

fontPath

사용자 정의 글꼴 파일의 위치를 지정합니다.

부모 요소:200페이지의 “글꼴”

자식 요소: 없음

내용
소스에 상대적인 사용자 정의 글꼴 파일의 경로입니다.

예제
```
<fontPath>ttf/space age.ttf</fontPath>
```

forceCPURenderModeForDevices

Adobe AIR 3.7 이상, iOS만 해당 - 선택 사항

지정한 장치 집합에 CPU 렌더링 모드를 강제 적용합니다. 이 기능은 선택적으로 나머지 iOS 장치에 GPU 렌더링 모드를 사용하려는 경우에 효과적입니다.

이 태그는 iPhone 태그의 자식으로 추가하고, 장치 모델 이름의 목록은 공백으로 구분하여 지정합니다. 유효한 장치 모델 이름으로는 다음이 포함됩니다.

iPad1,1	iPhone1,1	iPod1,1
iPad2,1	iPhone1,2	iPod2,1
iPad2,2	iPhone2,1	iPod3,3
iPad2,3	iPhone3.1	iPod4,1
iPad2,4	iPhone3,2	iPod5,1
iPad2,5	iPhone4,1	
iPad3,1	iPhone5,1	
iPad3,2		
iPad3,3		
iPad3,4		

부모 요소: 207페이지의 “iPhone”, 205페이지의 “initialWindow”

자식 요소: 없음

내용
공백으로 구분된 장치 모델 이름 목록입니다.

특성:
없음

예제
iOS:

```
...
<renderMode>GPU</renderMode>
...
<iPhone>
...
   <forceCPURenderModeForDevices>iPad1,1 iPhone1,1 iPhone1,2 iPod1,1
   </forceCPURenderModeForDevices>
</iPhone>
```

fullScreen

Adobe AIR 2.0 이상, iOS 및 Android - 선택 사항

응용 프로그램이 전체 화면 모드에서 시작되는지 여부를 지정합니다.

부모 요소:205페이지의 "initialWindow"

자식 요소: 없음

내용
true 또는 false(기본값)

예제

```
<fullscreen>true</fullscreen>
```

height

Adobe AIR 1.0 이상 - 선택 사항

응용 프로그램 기본 윈도우의 초기 높이입니다.

높이를 설정하지 않으면 루트 SWF 파일의 설정에 의해 결정되며 HTML 기반 AIR 응용 프로그램의 경우에는 운영 체제에 의해 결정됩니다.

AIR 2에서는 윈도우의 최대 높이가 2048픽셀에서 4096픽셀로 변경되었습니다.

부모 요소:205페이지의 "initialWindow"

자식 요소: 없음

내용
최대값이 4095인 양의 정수

예제

```
<height>4095</height>
```

icon

Adobe AIR 1.0 이상 - 선택 사항

icon 속성은 응용 프로그램에 사용될 하나 이상의 아이콘 파일을 지정합니다. 아이콘 포함은 선택 사항입니다. icon 속성을 지정하지 않으면 운영 체제가 기본 아이콘을 표시합니다.

지정된 경로는 응용 프로그램 루트 디렉토리를 기준으로 합니다. 아이콘 파일은 PNG 형식이어야 합니다. 모든 다음 아이콘 크기를 지정할 수 있습니다.

지정된 크기에 대한 요소가 있는 경우 파일의 이미지는 지정된 크기와 일치해야 합니다. 아무 크기도 제공되지 않은 경우에는 운영 체제에서 가장 근접한 크기가 아이콘의 지정된 용도에 맞게 조절됩니다.

참고: 지정된 아이콘은 AIR 패키지에 자동으로 추가되지 않으며 응용 프로그램이 패키지화될 때 아이콘 파일을 정확한 해당 상대적 위치에 포함해야 합니다.

최적의 결과를 얻으려면 사용 가능한 크기 각각에 대한 이미지를 제공하십시오. 또한 아이콘이 16비트 색상 모드와 32비트 색상 모드에서 제대로 표시되는지 확인하십시오.

부모 요소:188페이지의 “application”

자식 요소:204페이지의 “imageNxN”

내용

원하는 각 아이콘 크기에 대한 imageNxN 요소

예제

```
<icon>
    <image16x16>icons/smallIcon.png</image16x16>
    <image32x32>icons/mediumIcon.png</image32x32>
    <image48x48>icons/bigIcon.png</image48x48>
    <image128x128>icons/biggestIcon.png</image128x128>
</icon>
```

id

Adobe AIR 1.0 이상 - 필수 사항

응용 프로그램에 대한 식별자 문자열로, 응용 프로그램 ID라고 합니다. 역방향 DNS 스타일 식별자가 종종 사용되지만 이 스타일은 필수가 아닙니다.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용

ID 값은 다음 문자로 제한됩니다.

- 0–9
- a–z
- A–Z
- . (도트)
- -(하이픈)

이 값에는 1-212개의 문자를 포함해야 합니다. 이 요소는 필수 사항입니다.

예제

```
<id>org.example.application</id>
```

imageNxN

Adobe AIR 1.0 이상 - 선택 사항

응용 프로그램 디렉토리에 상대적인 아이콘 경로를 정의합니다.

각각 서로 다른 아이콘 크기를 지정하는 다음과 같은 아이콘 이미지를 사용할 수 있습니다.

- image16x16
- image29x29(AIR 2+)
- image32x32
- image36x36(AIR 2.5+)
- image48x48
- image50x50(AIR 3.4+)
- image57x57(AIR 2+)
- image58x58(AIR 3.4+)
- image72x72(AIR 2+)
- image100x100(AIR 3.4+)
- image114x114(AIR 2.6+)
- image128x128
- image144x144(AIR 3.4+)
- image512x512(AIR 2+)
- image1024x1024(AIR 3.4+)

아이콘은 image 요소가 나타내는 크기와 똑같은 PNG 그래픽이어야 합니다. 아이콘 파일은 응용 프로그램 패키지에 포함되어야 합니다. 응용 프로그램 설명자 문서에서 참조하는 아이콘이 자동으로 포함되지는 않습니다.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용

아이콘의 파일 경로에는 다양한 파일 시스템에서 파일 이름으로 사용할 수 없는 다음 문자를 제외한 모든 유니코드(UTF-8) 문자가 포함될 수 있습니다.

문자	16진수 코드
다양함	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E

문자	16진수 코드
?	x3F
\	x5C
|	x7C

예제

```
<image32x32>icons/icon32.png</image32x32>
```

InfoAdditions

Adobe AIR 1.0 이상 - 선택 사항

iOS 응용 프로그램에 대한 추가 속성을 지정할 수 있도록 합니다.

부모 요소:207페이지의 “iPhone”

자식 요소: iOS Info.plist 요소

내용

응용 프로그램의 Info.plist 설정으로 사용할 키-값 쌍을 지정하는 자식 요소가 포함됩니다. InfoAdditions 요소의 내용은 CDATA 블록으로 둘러싸여야 합니다.

키 값 쌍에 대한 자세한 내용 그리고 XML에서 키 값 쌍을 표현하는 방법에 대해서는 Apple iPhone 참조 라이브러리에서 정보 속성 목록 키 참조를 참조하십시오.

예제

```
<InfoAdditions>
    <![CDATA[
        <key>UIStatusBarStyle</key>
        <string>UIStatusBarStyleBlackOpaque</string>
        <key>UIRequiresPersistentWiFi</key>
        <string>NO</string>
    ]]>
</InfoAdditions>
```

기타 도움말 항목

74페이지의 “iOS 설정”

initialWindow

Adobe AIR 1.0 이상 - 필수 사항

기본 내용 파일 및 초기 응용 프로그램 모양을 정의합니다.

부모 요소:188페이지의 “application”

자식 요소: 다음과 같은 모든 요소가 initialWindow 요소의 자식으로 나타날 수 있습니다. 하지만 AIR가 플랫폼에서 윈도우를 지원하는지 여부에 따라 일부 요소는 무시됩니다.

요소	데스크톱	모바일
191페이지의 “aspectRatio”	무시됨	사용됨
191페이지의 “autoOrients”	무시됨	사용됨
193페이지의 “content”	사용됨	사용됨
194페이지의 “depthAndStencil”	사용됨	사용됨
202페이지의 “fullScreen”	무시됨	사용됨
202페이지의 “height”	사용됨	무시됨
210페이지의 “maximizable”	사용됨	무시됨
210페이지의 “maxSize”	사용됨	무시됨
210페이지의 “minimizable”	사용됨	무시됨
211페이지의 “minSize”	사용됨	무시됨
213페이지의 “renderMode”	사용됨(AIR 3.0 이상)	사용됨
214페이지의 “requestedDisplayResolution”	사용됨(AIR 3.6 이상)	무시됨
215페이지의 “resizable”	사용됨	무시됨
215페이지의 “softKeyboardBehavior”	무시됨	사용됨
217페이지의 “systemChrome”	사용됨	무시됨
218페이지의 “title”	사용됨	무시됨
218페이지의 “transparent”	사용됨	무시됨
219페이지의 “visible”	사용됨	무시됨
220페이지의 “width”	사용됨	무시됨
220페이지의 “x”	사용됨	무시됨
221페이지의 “y”	사용됨	무시됨

내용

응용 프로그램 모양 및 비헤이비어를 정의하는 자식 요소

예제

```
<initialWindow>
    <title>Hello World</title>
    <content>
        HelloWorld.swf
    </content>
    <depthAndStencil>true</depthAndStencil>
    <systemChrome>none</systemChrome>
    <transparent>true</transparent>
    <visible>true</visible>
    <maxSize>1024 800</maxSize>
    <minSize>320 240</minSize>
    <maximizable>false</maximizable>
    <minimizable>false</minimizable>
    <resizable>true</resizable>
    <x>20</x>
    <y>20</y>
    <height>600</height>
    <width>800</width>
    <aspectRatio>landscape</aspectRatio>
    <autoOrients>true</autoOrients>
    <fullScreen>false</fullScreen>
    <renderMode>direct</renderMode>
</initialWindow>
```

installFolder

Adobe AIR 1.0 이상 - 선택 사항

기본 설치 디렉토리의 하위 디렉토리를 식별합니다.

기본 설치 하위 디렉토리는 Windows에서는 Program Files 디렉토리이고, Mac OS에서는 /Applications 디렉토리입니다. Linux에서는 /opt/입니다. 예를 들어 installFolder 속성이 "Acme"로 설정되어 있고 응용 프로그램의 이름이 "ExampleApp"으로 지정되어 있는 경우, 응용 프로그램은 Windows에서는 C:\Program Files\Acme\ExampleApp에, Mac OS에서는 /Applications/Acme/Example.app에, Linux에서는 /opt/Acme/ExampleApp에 설치됩니다.

installFolder 속성은 선택 사항입니다. installFolder 속성을 지정하지 않으면 응용 프로그램이 name 속성을 기반으로 기본 설치 디렉토리의 하위 디렉토리에 설치됩니다.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용

installFolder 속성에는 다양한 파일 시스템에서 폴더 이름으로 사용할 수 없는 문자를 제외한 모든 유니코드(UTF-8) 문자가 포함될 수 있습니다. 예외 목록은 filename 속성을 참조하십시오.

중첩된 하위 디렉토리를 지정하려면 슬래시(/) 문자를 디렉토리 분리 기호 문자로 사용하십시오.

예제

```
<installFolder>utilities/toolA</installFolder>
```

iPhone

Adobe AIR 2.0, iOS만 해당 - 선택 사항

iOS 고유의 응용 프로그램 속성을 정의합니다.

부모 요소:188페이지의 “application”

자식 요소:

- 196페이지의 “Entitlements”
- 198페이지의 “externalSwfs”
- 201페이지의 “forceCPURenderModeForDevices”
- 205페이지의 “InfoAdditions”
- 214페이지의 “requestedDisplayResolution”

기타 도움말 항목

74페이지의 “iOS 설정”

manifest

Adobe AIR 2.5 이상, Android만 해당 - 선택 사항

응용 프로그램의 Android 매니페스트 파일에 추가할 정보를 지정합니다.

부모 요소:209페이지의 “manifestAdditions”

자식 요소: Android SDK에 의해 정의됩니다.

내용

manifest 요소는 엄밀히 말하자면 AIR 응용 프로그램 설명자 스킴에 포함되어 있지 않습니다. 오히려 Android 매니페스트 XML 문서의 루트입니다. manifest 요소 안에 넣는 내용은 AndroidManifest.xml 스킴을 준수해야 합니다. AIR 도구를 사용하여 APK 파일을 생성할 때 manifest 요소에 있는 정보는 응용 프로그램의 생성된 AndroidManifest.xml에서 그에 상응하는 부분에 복사됩니다.

AIR에서 직접 지원되는 것보다 최신 버전의 SDK에서만 사용할 수 있는 Android 매니페스트 값을 지정하는 경우 응용 프로그램을 패키지화할 때 -platformsdk 플래그를 ADT로 설정해야 합니다. 추가하려는 값을 지원하는 Android SDK 버전에 대한 파일 시스템 경로로 이 플래그를 설정합니다.

manifest 요소 자체는 AIR 응용 프로그램 설명자 내에서 CDATA 블록으로 둘러싸여야 합니다.

예제

```
<![CDATA[
    <manifest android:sharedUserID="1001">
        <uses-permission android:name="android.permission.CAMERA"/>
        <uses-feature android:required="false" android:name="android.hardware.camera"/>
        <application android:allowClearUserData="true"
                    android:enabled="true"
                    android:persistent="true"/>
    </manifest>
]]>
```

기타 도움말 항목

69페이지의 “Android 설정”

AndroidManifest.xml 파일

manifestAdditions

Adobe AIR 2.5 이상, Android만 해당

Android 매니페스트 파일에 추가할 정보를 지정합니다.

모든 Android 응용 프로그램에는 기본 응용 프로그램 속성을 정의하는 매니페스트 파일이 포함되어 있습니다. Android 매니페스트는 개념적으로 AIR 응용 프로그램 설명자와 비슷합니다. AIR for Android 응용 프로그램에는 응용 프로그램 설명자와 자동으로 생성된 Android 매니페스트 파일이 모두 들어 있습니다. AIR for Android 응용 프로그램이 패키지화될 때 이 manifestAdditions 요소에 있는 정보는 Android 매니페스트 문서의 상응하는 부분에 추가됩니다.

부모 요소:188페이지의 “android”

자식 요소:208페이지의 “manifest”

내용

manifestAdditions 요소에 있는 정보가 AndroidManifest XML 문서에 추가됩니다.

AIR는 응용 프로그램 및 런타임 기능이 올바르게 작동하도록 하기 위해 생성된 Android 매니페스트 문서에서 여러 매니페스트 항목을 설정합니다. 다음 설정은 무시할 수 없습니다.

manifest 요소의 다음 특성은 설정할 수 없습니다.

- package
- android:versionCode
- android:versionName

기본 activity 요소의 다음 특성은 설정할 수 없습니다.

- android:label
- android:icon

application 요소의 다음 특성은 설정할 수 없습니다.

- android:theme
- android:name
- android:label
- android:windowSoftInputMode
- android:configChanges
- android:screenOrientation
- android:launchMode

예제

```
<manifestAdditions>
    <![CDATA[
        <manifest android:installLocation="preferExternal">
            <uses-permission android:name="android.permission.INTERNET"/>
            <application android:allowClearUserData="true"
                        android:enabled="true"
                        android:persistent="true"/>
        </manifest>
    ]]>
</manifestAdditions>
```

기타 도움말 항목

69페이지의 “Android 설정”

AndroidManifest.xml 파일

maximizable

Adobe AIR 1.0 이상 - 선택 사항

윈도우를 최대화할 수 있는지 여부를 지정합니다.

참고: 윈도우 최대화가 크기 조절 작업인 Mac OS X와 같은 운영 체제에서 윈도우가 확대/축소 또는 크기 조절되지 않도록 하려면 maximizable과 resizable을 모두 false로 설정해야 합니다.

부모 요소:205페이지의 “initialWindow”

자식 요소: 없음

내용

true(기본값) 또는 false

예제

```
<maximizable>false</maximizable>
```

maxSize

Adobe AIR 1.0 이상 - 선택 사항

윈도우의 최대 크기입니다. 최대 크기를 설정하지 않으면 운영 체제에 의해 결정됩니다.

부모 요소:205페이지의 “initialWindow”

자식 요소: 없음

내용

최대 폭 및 높이를 나타내는 두 개의 정수로, 공백에 의해 분리됩니다.

참고: AIR 2에서는 AIR가 지원하는 최대 윈도우 크기가 2048x2048 픽셀에서 4096x4096 픽셀로 늘어났습니다. 스크린 좌표는 0부터 시작되므로 폭 또는 높이에 사용할 수 있는 최대값은 4095입니다.

예제

```
<maxSize>1024 360</maxSize>
```

minimizable

Adobe AIR 1.0 이상 - 선택 사항

윈도우를 최소화할 수 있는지 여부를 지정합니다.

부모 요소:205페이지의 “initialWindow”

자식 요소: 없음

내용
true(기본값) 또는 false

예제
```
<minimizable>false</minimizable>
```

minSize

Adobe AIR 1.0 이상 - 선택 사항

윈도우에 대해 허용되는 최소 크기를 지정합니다.

부모 요소:205페이지의 “initialWindow”

자식 요소: 없음

내용
최소 폭 및 높이를 나타내는 두 개의 정수로, 공백에 의해 분리됩니다. 운영 체제에서 정한 최소 크기가 응용 프로그램 설명자에 설정된 값보다 우선합니다.

예제
```
<minSize>120 60</minSize>
```

name

Adobe AIR 1.0 이상 - 선택 사항

AIR 응용 프로그램 설치 프로그램에서 표시하는 응용 프로그램 제목입니다.

name 요소가 지정되어 있지 않은 경우 AIR 응용 프로그램 설치 프로그램은 filename을 응용 프로그램 이름으로 표시합니다.

부모 요소:188페이지의 “application”

자식 요소:217페이지의 “text”

내용
다중 <text> 요소 대신 단일 텍스트 노드를 지정하는 경우 AIR 응용 프로그램 설치 프로그램은 시스템 언어와 상관없이 이 이름을 사용합니다.

AIR 1.0 응용 프로그램 설명자 스키마를 사용하면 이름에 대해 여러 text 요소가 아닌 하나의 간단한 텍스트 노드만 정의할 수 있습니다. AIR 1.1 이상에서는 name 요소에 다중 언어를 지정할 수 있습니다.

각 text 요소에 대한 xml:lang 특성은 RFC4646(http://www.ietf.org/rfc/rfc4646.txt)에 정의된 언어 코드를 지정합니다.

AIR 응용 프로그램 설치 프로그램은 사용되는 운영 체제의 사용자 인터페이스 언어에 가장 가까운 이름을 사용합니다. 예를 들어 응용 프로그램 설명자 파일의 name 요소에 en(영어) 로캘의 값이 포함된 설치를 살펴보면 운영 체제가 en(영어)을 사용자 인터페이스 언어로 식별하는 경우 AIR 응용 프로그램 설치 프로그램은 en 이름을 사용합니다. 시스템 사용자 인터페이스 언어가 en-US(미국 영어)인 경우에도 en 이름이 사용됩니다. 그러나 사용자 인터페이스 언어가 en-US이며 응용 프로그램 설명자 파일이 en-US 이름과 en-GB 이름을 모두 정의하는 경우 AIR 응용 프로그램 설치 프로그램은 en-US 값을 사용합니다. 응용 프로그램이 시스템 사용자 인터페이스 언어와 일치하는 이름을 정의하지 않는 경우 AIR 응용 프로그램 설치 프로그램은 응용 프로그램 설명자 파일에 정의된 첫 번째 name 값을 사용합니다.

name 요소는 AIR 응용 프로그램 설치 프로그램에서 사용되는 응용 프로그램 제목만 정의합니다. AIR 응용 프로그램 설치 프로그램은 중국어 번체, 중국어 간체, 체코어, 네덜란드어, 영어, 프랑스어, 독일어, 이탈리아어, 일본어, 한국어, 폴란드어, 포르투갈어(브라질), 러시아어, 스페인어, 스웨덴어, 터키어 등 다양한 언어를 지원합니다. AIR 응용 프로그램 설치 프로그램은 시스템 사용자 인터페이스 언어를 기반으로 응용 프로그램 제목 및 설명 외 텍스트에 대한 표시 언어를 선택합니다. 이러한 언어 선택 작업은 응용 프로그램 설명자 파일의 설정과는 별개로 수행됩니다.

name 요소는 설치되어 실행 중인 응용 프로그램에 대해 사용 가능한 로캘을 정의하지 **않습니다**. 다중 언어 응용 프로그램을 개발하는 방법에 대한 자세한 내용은 267페이지의 “AIR 응용 프로그램 지역화”를 참조하십시오.

예제

다음 예제에서는 간단한 텍스트 노드를 사용하여 이름을 정의합니다.

```
<name>Test Application</name>
```

AIR 1.1 이상에서 유효한 다음 예제는 <text> 요소 노드를 사용하여 세 개의 언어(영어, 프랑스 및 스페인어)로 이름을 지정합니다.

```
<name>
    <text xml:lang="en">Hello AIR</text>
    <text xml:lang="fr">Bonjour AIR</text>
    <text xml:lang="es">Hola AIR</text>
</name>
```

name

Adobe AIR 1.0 이상 - 필수 사항

파일 유형의 이름을 식별합니다.

부모 요소:199페이지의 “fileType”

자식 요소: 없음

내용

파일 유형의 이름을 나타내는 문자열입니다.

예제

```
<name>adobe.VideoFile</name>
```

programMenuFolder

Adobe AIR 1.0 이상 - 선택 사항

Windows 운영 체제의 [모든 프로그램] 메뉴 또는 Linux의 [응응 프로그램] 메뉴에서 응용 프로그램에 대한 단축키를 배치할 위치를 식별합니다. 현재 다른 운영 체제에서는 이 설정이 무시됩니다.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용

programMenuFolder 값에 사용되는 문자열에는 다양한 파일 시스템에서 폴더 이름으로 사용할 수 없는 문자를 제외한 모든 유니코드(UTF-8) 문자가 포함될 수 있습니다. 예외 목록은 filename 요소를 참조하십시오. 이 값의 마지막 문자로 슬래시(/) 문자를 사용하지 마십시오.

예제

```
<programMenuFolder>Example Company/Sample Application</programMenuFolder>
```

publisherID

Adobe AIR 1.5.3 이상 - 선택 사항

원래 AIR 버전 1.5.2 이하를 사용하여 만든 AIR 응용 프로그램을 업데이트하기 위한 제작자 ID를 식별합니다.

응용 프로그램 업데이트를 만들 때만 제작자 ID를 지정하십시오. publisherID 요소의 값은 이전 버전의 응용 프로그램에 대해 AIR에서 생성한 제작자 ID와 일치해야 합니다. 설치된 응용 프로그램의 경우 제작자 ID는 응용 프로그램이 설치된 폴더에 들어 있는 META-INF/AIR/publisherid 파일에서 확인할 수 있습니다.

AIR 1.5.3 이상을 사용하여 만든 새 응용 프로그램은 제작자 ID를 지정하지 않아야 합니다.

자세한 내용은 171페이지의 “AIR 제작자 ID”를 참조하십시오.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용

제작자 ID 문자열

예제

```
<publisherID>B146A943FBD637B68C334022D304CEA226D129B4.1</publisherID>
```

renderMode

Adobe AIR 2.0 이상 - 선택 사항

현재 컴퓨팅 장치에서 지원하는 경우 GPU(Graphics Processing Unit) 가속을 사용할 것인지 여부를 지정합니다.

부모 요소:205페이지의 “initialWindow”

자식 요소: 없음

내용

다음 값 중 하나:

- auto(기본값) - 현재 CPU 모드로 설정됩니다.
- cpu - 하드웨어 가속이 사용되지 않습니다.
- direct - 렌더링 컴포지션은 CPU에서 발생하고, 블록 전송 시에는 GPU가 사용됩니다. AIR 3 이상에서 사용 가능합니다.

  참고: 모바일 플랫폼용 AIR와 함께 Flash 내용의 GPU 가속 기능을 활용하려면 renderMode="gpu" 대신 renderMode="direct"(즉, Stage3D)를 사용하는 것이 좋습니다. Adobe에서는 Stage3D 기반 프레임워크인 Starling(2D)과 Away3D(3D)를 공식적으로 지원 및 권장합니다. Stage3D와 Starling/Away3D에 대한 자세한 내용은 http://gaming.adobe.com/getstarted/를 참조하십시오.
- gpu - 가능한 경우 하드웨어 가속이 사용됩니다.

  중요: Flex 응용 프로그램에 GPU 렌더링 모드를 사용하지 마십시오.

예제

```
<renderMode>direct</renderMode>
```

requestedDisplayResolution

iOS만 Adobe AIR 2.6 이상, OS X는 Adobe AIR 3.6 이상 - 선택 사항

고해상도 화면을 사용하는 장치 또는 컴퓨터 모니터에서 응용 프로그램이 표준 해상도를 사용할 것인지 아니면 고해상도를 사용할 것인지를 지정합니다. 기본값인 **standard**로 설정하면 해당 응용 프로그램이 표준 해상도 화면으로 나타납니다. **high**로 설정하면 응용 프로그램에서 각 고해상도 픽셀을 처리할 수 있습니다.

예를 들어 640x960 고해상도 iPhone 화면에서 설정이 **standard**인 경우 전체 화면 스테이지 크기는 320x480이고 각 응용 프로그램 픽셀은 네 개의 화면 픽셀을 사용하여 렌더링됩니다. 설정이 **high**이면 전체 화면 스테이지 크기는 640x960입니다.

표준 해상도 화면을 사용하는 장치에서는 어떤 설정을 사용하더라도 스테이지 크기가 화면 크기와 일치합니다.

requestedDisplayResolution 요소가 iPhone 요소에 중첩되어 있는 경우 iOS 장치에 적용됩니다. 이 경우 excludeDevices 특성을 사용하여 설정이 적용되지 않는 장치를 지정할 수 있습니다.

requestedDisplayResolution 요소가 initialWindow 요소에 중첩되어 있는 경우 고해상도 디스플레이를 지원하는 MacBook Pro 컴퓨터의 데스크톱 AIR 응용 프로그램에 적용됩니다. 지정된 값은 응용 프로그램에 사용되는 모든 기본 윈도우에 적용됩니다. requestedDisplayResolution 요소를 initialWindow 요소에 중첩하는 것은 AIR 3.6 이상에서 지원됩니다.

부모 요소: 207페이지의 “iPhone”, 205페이지의 “initialWindow”

자식 요소: 없음

내용

기본값인 **standard** 또는 **high**

특성:

excludeDevices - 공백으로 구분된 iOS 모델명 또는 모델명 접두어 목록입니다. 이를 통해 개발자가 어떤 장치는 고해상도를 사용하고 어떤 장치는 표준 해상도를 사용하도록 할 수 있습니다. 이 특성은 iOS에서만 사용할 수 있습니다 (requestedDisplayResolution 요소가 iPhone 요소에 중첩됨). excludeDevices 특성은 AIR 3.6 이상에서 사용할 수 있습니다.

모델명이 이 특성에 지정되어 있는 모든 장치는 requestedDisplayResolution 값이 지정된 값의 반대입니다. 즉, requestedDisplayResolution 값이 **high**이면 예외 장치는 표준 해상도를 사용합니다. requestedDisplayResolution 값이 **standard**이면 예외 장치는 고해상도를 사용합니다.

값은 iOS 장치 모델명 또는 모델명 접두어입니다. 예를 들어 값 iPad3.1은 구체적으로 Wi-Fi 3세대 iPad를 가리킵니다(그러나 GSM 또는 CDMA 3세대 iPad는 해당되지 않음). 대신 값 iPad3은 모든 3세대 iPad를 가리킵니다. 비공식 iOS 모델명 목록은 iPhone wiki 모델 페이지에서 확인할 수 있습니다.

예제

데스크톱:

```
<initialWindow>
    <requestedDisplayResolution>high</requestedDisplayResolution>
</initialWindow>
```

iOS:

```
<iPhone>
    <requestedDisplayResolution excludeDevices="iPad3 iPad4">high</requestedDisplayResolution>
</iPhone>
```

resizable

Adobe AIR 1.0 이상 - 선택 사항

윈도우 크기를 조절할 수 있는지 여부를 지정합니다.

참고: 윈도우 최대화가 크기 조절 작업인 Mac OS X와 같은 운영 체제에서 윈도우가 확대/축소 또는 크기 조절되지 않도록 하려면 maximizable과 resizable을 모두 false로 설정해야 합니다.

부모 요소:205페이지의 "initialWindow"

자식 요소:

내용
true(기본값) 또는 false

예제
```
<resizable>false</resizable>
```

softKeyboardBehavior

Adobe AIR 2.6 이상, 모바일 프로파일 - 선택 사항

가상 키보드가 표시될 때 응용 프로그램의 기본 비헤이비어를 지정합니다. 기본 비헤이비어는 응용 프로그램을 위로 이동하는 것입니다. 런타임은 포커스가 있는 텍스트 필드 또는 대화형 객체를 화면에 계속 표시합니다. 응용 프로그램에서 자체적인 키보드 처리 논리를 제공하지 않는 경우 **pan** 옵션을 사용하십시오.

softKeyboardBehavior 요소를 **none**로 설정함으로써 자동 비헤이비어를 끌 수도 있습니다. 이러한 경우에는 소프트 키보드가 표시될 때 텍스트 필드 및 대화형 객체가 SoftKeyboardEvent를 전달하지만 런타임이 응용 프로그램을 이동하거나 응용 프로그램의 크기를 조절하지 않습니다. 텍스트 입력 영역이 눈에 보이도록 유지하는 동작은 응용 프로그램에서 수행합니다.

부모 요소:205페이지의 "initialWindow"

자식 요소: 없음

내용
none 또는 **pan**. 기본값은 **pan**입니다.

예제
```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

기타 도움말 항목
SoftKeyboardEvent

supportedLanguages

Adobe AIR 3.2 이상 - 선택 사항

응용 프로그램에서 지원하는 언어를 식별합니다. 이 요소는 iOS, Mac 전용 런타임 및 Android 응용 프로그램에서만 사용됩니다. 다른 모든 응용 프로그램 유형에서는 이 요소가 무시됩니다.

이 요소를 지정하지 않을 경우 기본적으로 패키지 프로그램은 응용 프로그램 유형에 따라 다음 작업을 수행합니다.

- iOS - AIR 런타임이 지원하는 모든 언어가 iOS App Store에서 응용 프로그램의 지원되는 언어로 나열됩니다.

- Mac 전용 런타임 - 전용 번들과 함께 패키지화된 응용 프로그램에 지역화 정보가 없습니다.
- Android - 응용 프로그램 번들에 AIR 런타임이 지원하는 모든 언어에 대한 리소스가 있습니다.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용

공백으로 구분된 지원되는 언어 목록입니다. 유효한 언어 값은 AIR 런타임이 지원하는 언어의 ISO 639-1 값인 en, de, es, fr, it, ja, ko, pt, ru, cs, nl, pl, sv, tr, zh, da, nb, iw입니다.

<supportedLanguages> 요소의 값이 비어 있으면 패키지 프로그램에서 오류가 생성됩니다.

참고: <supportedLanguages> 태그를 사용하는 상황에서 이 태그에 특정 언어가 포함되어 있지 않으면 지역화된 태그(예: 이름 태그)가 해당 언어 값을 무시합니다. 기본 확장에 <supportedLangauges> 태그를 통해 지정되지 않은 언어에 대한 리소스가 있는 경우 경고가 발생하고 해당 언어에 대해 리소스가 무시됩니다.

예제

```
<supportedLanguages>en ja fr es</supportedLanguages>
```

supportedProfiles

Adobe AIR 2.0 이상 - 선택 사항

응용 프로그램에서 지원하는 프로파일을 식별합니다.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용

supportedProfiles 요소에는 다음 값 중 하나를 사용할 수 있습니다.

- desktop - 데스크톱 프로파일은 AIR 파일을 사용하여 데스크톱 컴퓨터에 설치되는 AIR 응용 프로그램을 위한 것입니다. 이러한 응용 프로그램은 기본 응용 프로그램과의 통신을 제공하는 NativeProcess 클래스에 액세스할 수 없습니다.
- extendedDesktop - 확장 데스크톱 프로파일은 기본 응용 프로그램 설치 프로그램을 사용하여 데스크톱 컴퓨터에 설치되는 AIR 응용 프로그램을 위한 것입니다. 이러한 응용 프로그램은 기본 응용 프로그램과의 통신을 제공하는 NativeProcess 클래스에 액세스할 수 있습니다.
- mobileDevice - 휴대 장치 프로파일은 모바일 응용 프로그램을 위한 것입니다.
- extendedMobileDevice - 확장 휴대 장치 프로파일은 현재 사용되고 있지 않습니다.

supportedProfiles 속성은 선택 사항입니다. 응용 프로그램 설명자 파일에 이 요소를 포함하지 않으면 모든 프로파일에서 응용 프로그램을 컴파일하고 배포할 수 있습니다.

프로파일을 여러 개 지정하려면 공백 문자로 프로파일을 구분하십시오. 예를 들어, 다음 설정은 응용 프로그램을 데스크톱 및 확장된 프로파일에서만 사용할 수 있도록 지정합니다.

```
<supportedProfiles>desktop extendedDesktop</supportedProfiles>
```

참고: ADL을 사용하여 응용 프로그램을 실행하고 ADL -profile 옵션에 대한 값을 지정하지 않으면 응용 프로그램 설명자에 있는 첫 번째 프로파일이 사용됩니다. 응용 프로그램 설명자에 프로파일이 지정되어 있지 않으면 데스크톱 프로파일이 사용됩니다.

예제

```
<supportedProfiles>desktop mobileDevice</supportedProfiles>
```

기타 도움말 항목

222페이지의 “장치 프로파일”

68페이지의 “지원되는 프로파일”

systemChrome

Adobe AIR 1.0 이상 - 선택 사항

초기 응용 프로그램 윈도우가 운영 체제에서 제공하는 표준 제목 표시줄, 테두리 및 컨트롤과 함께 생성되는지 여부를 지정합니다.

런타임에는 윈도우의 시스템 크롬 설정을 변경할 수 없습니다.

부모 요소:205페이지의 “initialWindow”

자식 요소: 없음

내용

다음 값 중 하나:

- none - 시스템 크롬이 제공되지 않습니다. 응용 프로그램(또는 Flex와 같은 응용 프로그램 프레임워크)에서 윈도우 크롬을 정하여 표시해야 합니다.
- 표준(기본값) - 운영 체제에서 시스템 크롬을 제공합니다.

예제

```
<systemChrome>standard</systemChrome>
```

text

Adobe AIR 1.1 이상 - 선택 사항

지역화된 문자열을 지정합니다.

text 요소의 xml:lang 특성은 RFC4646(http://www.ietf.org/rfc/rfc4646.txt)에 정의된 언어 코드를 지정합니다.

AIR 응용 프로그램 설치 프로그램은 사용되는 운영 체제의 사용자 인터페이스 언어에 가장 가까운 xml:lang 특성 값의 text 요소를 사용합니다.

예를 들어 text 요소에 en(영어) 로캘에 대한 값이 포함되어 있는 설치를 가정해 보십시오. 운영 체제가 en(영어)을 사용자 인터페이스 언어로 식별하는 경우 AIR 응용 프로그램 설치 프로그램은 en 이름을 사용합니다. 시스템 사용자 인터페이스 언어가 en-US(미국 영어)인 경우에도 en 이름이 사용됩니다. 그러나 사용자 인터페이스 언어가 en-US이며 응용 프로그램 설명자 파일이 en-US 이름과 en-GB 이름을 모두 정의하는 경우 AIR 응용 프로그램 설치 프로그램은 en-US 값을 사용합니다.

응용 프로그램이 시스템 사용자 인터페이스 언어와 일치하는 text 요소를 정의하지 않는 경우 AIR 응용 프로그램 설치 프로그램은 응용 프로그램 설명자 파일에 정의된 첫 번째 name 값을 사용합니다.

부모 요소:

- 211페이지의 “name”
- 195페이지의 “description”

자식 요소: 없음

내용

지역화된 텍스트의 로캘 및 문자열을 지정하는 xml:lang 특성

예제

```
<text xml:lang="fr">Bonjour AIR</text>
```

title

Adobe AIR 1.0 이상 - 선택 사항

초기 응용 프로그램 윈도우의 제목 표시줄에 표시되는 제목을 지정합니다.

systemChrome 요소가 standard로 설정된 경우에만 제목이 표시됩니다.

부모 요소:205페이지의 “initialWindow”

자식 요소: 없음

내용

윈도우 제목을 포함하는 문자열

예제

```
<title>Example Window Title</title>
```

transparent

Adobe AIR 1.0 이상 - 선택 사항

초기 응용 프로그램 윈도우가 데스크톱과 알파-블렌딩되는지 여부를 지정합니다.

투명도를 사용하는 윈도우는 보다 느리게 그려지며 더 많은 메모리가 필요할 수 있습니다. 런타임에는 투명 설정을 변경할 수 없습니다.

중요: systemChrome이 none인 경우에만 transparent를 true로 설정할 수 있습니다.

부모 요소:205페이지의 “initialWindow”

자식 요소: 없음

내용

true 또는 false(기본값)

예제

```
<transparent>true</transparent>
```

version

Adobe AIR 1.0부터 2.0까지 - 필수 사항, AIR 2.5 이상에서는 허용되지 않음

응용 프로그램에 대한 버전 정보를 지정합니다.

버전 문자열은 응용 프로그램이 정의하는 지정자입니다. AIR는 어떤 방식으로도 버전 문자열을 해석하지 않습니다. 따라서 "3.0" 버전이 "2.0" 버전보다 최신 버전이라고 가정할 수 없습니다. 몇 가지 예로는 "1.0", ".4", "0.5", "4.9", "1.3.4a" 등을 들 수 있습니다.

AIR 2.5 이상에서는 version 요소가 versionNumber 및 versionLabel 요소로 대체됩니다.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용
응용 프로그램 버전을 포함하는 문자열

예제
```
<version>0.1 Alpha</version>
```

versionLabel

Adobe AIR 2.5 이상 - 선택 사항

사람이 읽을 수 있는 버전의 문자열을 지정합니다.

설치 대화 상자에서 versionNumber 요소의 값 대신 버전 레이블의 값이 표시됩니다. versionLabel이 사용되지 않으면 둘 다에 대해 versionNumber가 사용됩니다.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용
공개적으로 표시되는 버전 텍스트를 포함하는 문자열

예제
```
<versionLabel>0.9 Beta</versionlabel>
```

versionNumber

Adobe AIR 2.5 이상 - 필수 사항

응용 프로그램 버전 번호입니다.

부모 요소:188페이지의 “application”

자식 요소: 없음

내용
버전 번호는 마침표로 분리된 최대 세 개의 정수를 포함할 수 있습니다. 각 정수는 0에서 999 사이(포함)의 숫자여야 합니다.

예제
```
<versionNumber>1.0.657</versionNumber>

<versionNumber>10</versionNumber>

<versionNumber>0.01</versionNumber>
```

visible

Adobe AIR 1.0 이상 - 선택 사항

초기 응용 프로그램 윈도우가 생성되는 즉시 표시되는지 여부를 지정합니다.

초기 윈도우를 비롯한 AIR 윈도우는 기본적으로 숨겨진 상태로 생성됩니다. NativeWindow 객체의 activate() 메서드를 호출하거나 visible 속성을 true로 설정하면 윈도우를 표시할 수 있습니다. 윈도우 위치, 윈도우 크기 및 윈도우 내용의 레이아웃에 대한 변경 사항이 표시되지 않도록 처음에는 기본 윈도우를 숨겨 둘 수도 있습니다.

MXML 정의에서 visible 특성이 false로 설정되어 있지 않은 한 Flex mx:WindowedApplication 구성 요소는 applicationComplete 이벤트가 전달되기 직전에 윈도우를 자동으로 표시 및 활성화합니다.

윈도우를 지원하지 않는 모바일 프로파일의 장치에서는 visible 설정이 무시됩니다.

부모 요소:205페이지의 "initialWindow"

자식 요소: 없음

내용
true 또는 false(기본값)

예제
```
<visible>true</visible>
```

width

Adobe AIR 1.0 이상 - 선택 사항

응용 프로그램 기본 윈도우의 초기 폭입니다.

폭을 설정하지 않으면 루트 SWF 파일의 설정에 의해 결정되며 HTML 기반 AIR 응용 프로그램의 경우에는 운영 체제에 의해 결정됩니다.

AIR 2에서는 윈도우의 최대 폭이 2048픽셀에서 4096픽셀로 변경되었습니다.

부모 요소:205페이지의 "initialWindow"

자식 요소: 없음

내용
최대값이 4095인 양의 정수

예제
```
<width>1024</width>
```

x

Adobe AIR 1.0 이상 - 선택 사항

초기 응용 프로그램 윈도우의 가로 위치입니다.

대부분의 경우 고정된 값을 할당하는 것보다 운영 체제에서 윈도우의 초기 위치를 결정하도록 하는 것이 좋습니다.

화면 좌표계의 원점(0,0)은 기본 데스크톱 화면(운영 체제에 의해 결정됨)의 왼쪽 위 모서리입니다.

부모 요소:205페이지의 "initialWindow"

자식 요소: 없음

내용
정수 값입니다.

예제

```
<x>120</x>
```

y

Adobe AIR 1.0 이상 - 선택 사항

초기 응용 프로그램 윈도우의 세로 위치입니다.

대부분의 경우 고정된 값을 할당하는 것보다 운영 체제에서 윈도우의 초기 위치를 결정하도록 하는 것이 좋습니다.

화면 좌표계의 원점(0,0)은 기본 데스크톱 화면(운영 체제에 의해 결정됨)의 왼쪽 위 모서리입니다.

부모 요소:205페이지의 “initialWindow”

자식 요소: 없음

내용

정수 값입니다.

예제

```
<y>250</y>
```

15장: 장치 프로파일

Adobe AIR 2 이상

프로파일은 응용 프로그램이 작동하는 컴퓨팅 장치의 클래스를 정의하기 위한 메커니즘입니다. 프로파일은 특정 클래스의 장치에서 일반적으로 지원되는 API 및 기능 집합을 정의합니다. 사용 가능한 프로파일은 다음과 같습니다.

- desktop
- extendedDesktop
- mobileDevice
- extendedMobileDevice

응용 프로그램 설명자에서 응용 프로그램의 프로파일을 정의할 수 있습니다. 포함된 프로파일에 있는 컴퓨터 및 장치의 사용자는 응용 프로그램을 설치할 수 있지만 다른 컴퓨터 및 장치의 사용자는 설치할 수 없습니다. 예를 들어 응용 프로그램 설명자에 데스크톱 프로파일만 포함하는 경우에는 데스크톱 컴퓨터에서만 사용자가 응용 프로그램을 설치하고 실행할 수 있습니다.

응용 프로그램에서 제대로 지원하지 않는 프로파일을 포함하면 해당 환경에서 사용자가 작업하기가 어려울 수 있습니다. 응용 프로그램 설명자에서 프로파일을 지정하지 않으면 AIR에서 응용 프로그램을 제한하지 않습니다. 지원되는 포맷 중 하나로 응용 프로그램을 패키지화하면 모든 프로파일에 있는 장치의 사용자가 해당 응용 프로그램을 설치할 수 있습니다. 하지만 런타임에 제대로 작동하지 않을 수 있습니다.

해당되는 경우 응용 프로그램을 패키지화할 때 프로파일 제한이 적용됩니다. 예를 들어 extendedDesktop 프로파일만 포함하면 응용 프로그램을 AIR 파일로 패키지화할 수 없고 기본 설치 프로그램으로만 패키지화할 수 있습니다. 마찬가지로 mobileDevice 프로파일을 포함하지 않으면 응용 프로그램을 Android APK로 패키지화할 수 없습니다.

한 컴퓨팅 장치에서 여러 개의 프로파일을 지원할 수 있습니다. 예를 들어 데스크톱 컴퓨터의 AIR는 데스크톱 및 extendedDesktop 프로파일에 있는 응용 프로그램을 지원합니다. 하지만 확장 데스크톱 프로파일 응용 프로그램은 기본 프로세스와만 통신할 수 있으며 기본 설치 프로그램(exe, dmg, deb 또는 rpm)으로 패키지화해야 합니다. 반면에 데스크톱 프로파일 응용 프로그램은 기본 프로세스와 통신할 수 없습니다. 데스크톱 프로파일 응용 프로그램은 AIR 파일 또는 기본 설치 프로그램으로 패키지화할 수 있습니다.

프로파일에 기능을 포함하면 해당 프로파일이 정의되는 장치 클래스에서 이 기능이 공통적으로 지원됩니다. 그렇다고 해서 프로파일에 있는 모든 장치가 모든 기능을 지원하는 것은 아닙니다. 예를 들어 대부분의 휴대폰에 Accelerometer가 포함되어 있지만 모든 휴대폰에 포함된 것은 아닙니다. 범용 지원이 없는 클래스 및 기능에는 주로 부울 속성이 있으며 기능을 사용하기 전에 이 부울 속성을 확인할 수 있습니다. 예를 들어 Accelerometer의 경우 정적 속성 Accelerometer.isSupported를 테스트하여 현재 장치에 지원되는 Accelerometer가 있는지 여부를 확인할 수 있습니다.

응용 프로그램 설명자에서 supportedProfiles 요소를 사용하여 다음 프로파일을 AIR 응용 프로그램에 할당할 수 있습니다.

데스크톱 데스크톱 프로파일은 데스크톱 컴퓨터에 AIR 파일로 설치되는 AIR 응용 프로그램의 기능 세트를 정의합니다. 이러한 응용 프로그램은 지원되는 데스크톱 플랫폼(Mac OS, Windows 및 Linux)에서 설치되어 실행됩니다. AIR 2 이전 버전에서 개발된 AIR 응용 프로그램은 데스크톱 프로파일로 간주될 수 있습니다. 일부 API는 이 프로파일에서 작동하지 않습니다. 예를 들어, 데스크톱 응용 프로그램이 기본 프로세스와 통신할 수 없습니다.

확장 데스크톱 확장 데스크톱 프로파일은 기본 설치 프로그램으로 패키지화되어 설치된 AIR 응용 프로그램의 기능 집합을 정의합니다. 이러한 기본 설치 프로그램은 Windows의 경우 EXE 파일, Mac OS의 경우 DMG 파일, Linux의 경우 BIN, DEB 또는 RPM 파일입니다. 확장 데스크톱 응용 프로그램에는 데스크톱 프로파일 응용 프로그램에 제공되지 않는 추가 기능이 있습니다. 자세한 내용은 51페이지의 “데스크톱 기본 설치 프로그램 패키지화”를 참조하십시오.

휴대 장치 휴대 장치 프로파일은 휴대 전화, 타블렛과 같은 휴대 장치에 설치된 응용 프로그램의 기능 집합을 정의합니다. 이러한 응용 프로그램은 Android, Blackberry Tablet OS 및 iOS를 포함하여 지원되는 모바일 플랫폼에서 설치 및 실행됩니다.

확장 휴대 장치 확장 휴대 장치 프로파일은 휴대 장치에 설치된 응용 프로그램의 확장된 기능 집합을 정의합니다. 현재로서 이 프로파일을 지원하는 장치는 없습니다.

응용 프로그램 설명자 파일에 대상 프로파일 제한

Adobe AIR 2 이상

AIR 2부터 응용 프로그램 설명자 파일에는 대상 프로파일을 제한하는 데 사용할 수 있는 supportedProfiles 요소가 포함됩니다. 예를 들어, 다음 설정은 응용 프로그램을 데스크톱 프로파일에서만 사용할 수 있도록 지정합니다.

```
<supportedProfiles>desktop</supportedProfiles>
```

이 요소가 설정되면 사용자가 나열한 프로파일에서만 응용 프로그램을 패키지화할 수 있습니다. 다음 값을 사용합니다.

- desktop - 데스크톱 프로파일
- extendedDesktop - 확장 데스크톱 프로파일
- mobileDevice - 휴대 장치 프로파일

supportedProfiles 요소는 선택 사항입니다. 응용 프로그램 설명자 파일에 이 요소를 포함하지 않으면 모든 프로파일에서 응용 프로그램을 패키지화하고 배포할 수 있습니다.

supportedProfiles 요소에서 여러 프로파일을 지정하려면 다음과 같이 공백 문자를 사용하여 구분하십시오.

```
<supportedProfiles>desktop extendedDesktop</supportedProfiles>
```

각 프로파일의 기능

Adobe AIR 2 이상

다음 표에는 일부 프로파일에서만 지원되는 클래스 및 기능이 나와 있습니다.

클래스 또는 기능	desktop	extendedDesktop	mobileDevice
Accelerometer(Accelerometer.isSupported)	아니요	아니요	확인
Accessibility(Capabilities.hasAccessibility)	예	예	아니요
어쿠스틱 에코 제거 (Microphone.getEnhancedMicrophone())	예	예	아니요
ActionScript 2	예	예	아니요
CacheAsBitmap 매트릭스	아니요	아니요	예
Camera(Camera.isSupported)	예	예	예
CameraRoll	아니요	아니요	예
CameraUI(CameraUI.isSupported)	아니요	아니요	예
전용 런타임 번들	예	예	예
ContextMenu(ContextMenu.isSupported)	예	예	아니요

클래스 또는 기능	desktop	extendedDesktop	mobileDevice
DatagramSocket(DatagramSocket.isSupported)	예	예	예
DockIcon(NativeApplication.supportsDockIcon)	확인	확인	아니요
드래그 앤 드롭 (NativeDragManager.isSupported)	예	예	확인
EncyptedLocalStore(EncyptedLocalStore.isSupported)	예	예	예
Flash Access(DRMManager.isSupported)	예	예	아니요
GameInput(GameInput.isSupported)	아니요	아니요	아니요
Geolocation(Geolocation.isSupported)	아니요	아니요	확인
HTMLLoader(HTMLLoader.isSupported)	예	예	아니요
IME(IME.isSupported)	예	예	확인
LocalConnection(LocalConnection.isSupported)	예	예	아니요
Microphone(Microphone.isSupported)	예	예	확인
다중 채널 오디오 (Capabilities.hasMultiChannelAudio())	아니요	아니요	아니요
기본 확장	아니요	예	예
NativeMenu(NativeMenu.isSupported)	예	예	아니요
NativeProcess(NativeProcess.isSupported)	아니요	예	아니요
NativeWindow(NativeWindow.isSupported)	예	예	아니요
NetworkInfo(NetworkInfo.isSupported)	예	예	확인
기본 응용 프로그램으로 파일 열기	제한	예	아니요
PrintJob(PrintJob.isSupported)	예	예	아니요
SecureSocket(SecureSocket.isSupported)	예	예	확인
ServerSocket(ServerSocket.isSupported)	예	예	예
Shader	예	예	제한
Stage3D(Stage.stage3Ds.length)	예	예	예
스테이지 방향 (Stage.supportsOrientationChange)	아니요	아니요	예
StageVideo	아니요	아니요	확인
StageWebView(StageWebView.isSupported)	예	예	예
로그인할 때 응용 프로그램 시작 (NativeApplication.supportsStartAtLogin)	예	예	아니요

클래스 또는 기능	desktop	extendedDesktop	mobileDevice
StorageVolumeInfo(StorageVolumeInfo.isSupported)	예	예	아니요
시스템 유휴 모드	아니요	아니요	예
SystemTrayIcon(NativeApplication.supportsSystemTrayIcon)	확인	확인	아니요
Text Layout Framework 입력	예	예	아니요
Updater(Updater.isSupported)	예	아니요	아니요
XMLSignatureValidator(XMLSignatureValidator.isSupported)	예	예	아니요

표에 있는 항목의 의미는 다음과 같습니다.

- **확인** - 프로파일에 있는 일부 장치에서 기능이 지원되지만 모든 장치에서 지원되는 것은 아닙니다. 사용하기 전에 런타임에 기능이 지원되는지 여부를 확인해야 합니다.
- **제한** - 기능이 지원되지만 크게 제한됩니다. 자세한 내용은 관련 설명서를 참조하십시오.
- **아니요** - 프로파일에서 기능이 지원되지 않습니다.
- **예** - 프로파일에서 기능이 지원됩니다. 개별 컴퓨팅 장치에서 기능에 필요한 하드웨어가 없을 수도 있습니다. 예를 들어 모든 휴대폰에 카메라가 있는 것은 아닙니다.

ADL로 디버깅할 때 프로파일 지정

Adobe AIR 2 이상

ADL은 응용 프로그램 설명자 파일의 supportedProfiles 요소에서 지원되는 프로파일이 지정되었는지 확인합니다. 파일이 지정되었으면 디버그할 때 기본적으로 ADL은 프로파일로 나열된 것 중 지원되는 첫 번째 프로파일을 사용합니다.

-profile 명령줄 인수를 사용하여 ADL 디버그 세션에 대한 프로파일을 지정할 수 있습니다. 144페이지의 “ADL(AIR Debug Launcher)”을 참조하십시오. 응용 프로그램 설명자 파일에서 supportedProfiles 요소에 프로파일을 지정하는지 여부에 관계없이 이 인수를 사용할 수 있습니다. 하지만 supportedProfiles 요소를 지정할 경우 명령줄에 지정한 프로파일을 포함해야 합니다. 그렇지 않으면 ADL에서 오류가 발생합니다.

16장: AIR.SWF 인 브라우저 API

연속 설치 badge.swf 사용자 정의

SDK와 함께 제공되는 badge.swf 파일을 사용하지 않고 고유 SWF 파일을 만들어 브라우저 페이지에 사용할 수도 있습니다. 사용자 정의 SWF 파일은 다음과 같은 방법으로 런타임과 상호 작용할 수 있습니다.

- AIR 응용 프로그램 설치. 231페이지의 "브라우저에서 AIR 응용 프로그램 설치"를 참조하십시오.
- 특정 AIR 응용 프로그램이 설치되어 있는지 확인. 230페이지의 "AIR 응용 프로그램이 설치되어 있는지 웹 페이지에서 확인"을 참조하십시오.
- 런타임이 설치되어 있는지 확인. 230페이지의 "런타임이 설치되어 있는지 확인"을 참조하십시오.
- 사용자 시스템에 설치된 AIR 응용 프로그램 시작. 자세한 내용은 232페이지의 "브라우저에서 설치된 AIR 응용 프로그램 시작"을 참조하십시오.

이러한 모든 기능은 adobe.com에 있는 SWF 파일(air.swf)의 API를 호출하여 제공됩니다. badge.swf 파일을 사용자 정의하여 사용자의 SWF 파일에서 air.swf API를 호출할 수 있습니다.

브라우저에서 실행되는 SWF 파일은 LocalConnection 클래스를 사용하여 실행 중인 AIR 응용 프로그램과 통신할 수도 있습니다. 자세한 내용은 다른 Flash Player 및 AIR 인스턴스와의 통신(ActionScript 개발자용) 또는 Communicating with other Flash Player and AIR instances(HTML 개발자용)을 참조하십시오.

중요: 이 단원에 설명된 기능 및 air.swf 파일의 API를 사용하려면 최종 사용자의 Windows 또는 Mac OS 웹 브라우저에 Adobe® Flash® Player 9 업데이트 3 이상이 설치되어 있어야 합니다. Linux에서 연속 설치 기능을 사용하려면 Flash Player 10 버전 10,0,12,36 이상이 필요합니다. 코드를 작성하여 Flash Player의 설치된 버전을 확인하고 필요한 버전의 Flash Player가 설치되어 있지 않은 경우 사용자에게 대체 인터페이스를 제공할 수 있습니다. 예를 들어 이전 버전의 Flash Player가 설치되어 있는 경우 badge.swf 파일이나 air.swf API를 사용하여 응용 프로그램을 설치하는 대신 다운로드 버전의 AIR 파일에 대한 링크를 제공할 수 있습니다.

badge.swf 파일을 사용하여 AIR 응용 프로그램 설치

연속 설치 기능을 쉽게 사용할 수 있게 해주는 badge.swf 파일이 AIR SDK 및 Flex SDK에 포함되어 있습니다. badge.swf는 웹 페이지의 링크에서 런타임 및 AIR 응용 프로그램을 설치할 수 있습니다. 웹 사이트에 배포할 수 있도록 badge.swf 파일 및 소스 코드가 제공됩니다.

웹 페이지에 badge.swf 파일 포함

1. AIR SDK 또는 Flex SDK의 samples/badge 디렉토리에 제공되는 다음 파일을 찾아 웹 서버에 추가합니다.
   - badge.swf
   - default_badge.html
   - AC_RunActiveContent.js
2. 텍스트 편집기에서 default_badge.html 페이지를 엽니다.
3. default_badge.html 페이지의 AC_FL_RunContent() JavaScript 함수에서 다음에 대해 FlashVars 매개 변수 정의를 조정합니다.

매개 변수	설명
appname	런타임이 설치되어 있지 않을 때 SWF 파일에서 표시하는 응용 프로그램의 이름입니다.
appurl	(필수) 다운로드할 AIR 파일의 URL입니다. 상대 URL이 아닌 절대 URL을 사용해야 합니다.
airversion	(필수) 런타임 버전 1.0의 경우 이 값을 1.0으로 설정합니다.
imageurl	배지에 표시할 이미지의 URL(선택 사항)입니다.
buttoncolor	다운로드 버튼의 색입니다(FFCC00과 같은 16진수 값으로 지정).
messagecolor	런타임이 설치되어 있지 않을 때 버튼 아래에 표시되는 텍스트 메시지의 색입니다(FFCC00과 같은 16진수 값으로 지정).

4 badge.swf 파일의 최소 크기는 217 x 180픽셀입니다. 필요에 맞게 AC_FL_RunContent() 함수의 width 및 height 매개 변수의 값을 조정합니다.

5 default_badge.html 파일의 이름을 변경하고 필요에 맞게 코드를 조정하거나 다른 HTML 페이지에 포함합니다.

참고: badge.swf 파일을 로드하는 HTML embed 태그에서 wmode 특성을 설정하면 안 됩니다. 기본 설정인 "window"를 유지하십시오. 다른 wmode 설정을 사용하면 일부 시스템에서 설치가 되지 않습니다. 또한 다른 wmode 설정을 사용하면 "오류 #2044: 처리되지 않은 ErrorEvent:. text=Error #2074: 스테이지가 너무 작아 다운로드 UI에 맞지 않습니다." 오류가 생성됩니다.

badge.swf 파일을 편집하여 다시 컴파일할 수도 있습니다. 자세한 내용은 228페이지의 “badge.swf 파일 수정”을 참조하십시오.

웹 페이지의 연속 설치 링크에서 AIR 응용 프로그램 설치

페이지에 연속 설치 링크가 추가되면 사용자는 SWF 파일의 링크를 클릭하여 AIR 응용 프로그램을 설치할 수 있습니다.

1 Flash Player 버전 9 업데이트 3 이상(Windows 및 Mac OS) 또는 버전 10(Linux)이 설치된 웹 브라우저에서 HTML 페이지로 이동합니다.

2 웹 페이지에서 badge.swf 파일의 링크를 클릭합니다.

- 런타임을 설치한 경우 다음 단계로 건너뜁니다.
- 런타임을 설치하지 않은 경우 런타임을 설치할 것인지 여부를 묻는 대화 상자가 표시됩니다. 런타임을 설치(2페이지의 “Adobe AIR 설치” 참조)하고 다음 단계를 진행합니다.

3 [설치] 윈도우에서 기본 설정을 선택한 상태로 두고 [계속]을 클릭합니다.

Windows 컴퓨터에서 AIR는 자동으로 다음을 수행합니다.

- 응용 프로그램을 c:\Program Files\에 설치
- 응용 프로그램에 대한 바탕 화면 단축키 만들기
- [시작] 메뉴 단축키 만들기
- [제어판]의 [프로그램 추가/제거]에서 응용 프로그램에 대한 항목 추가

Mac OS에서 설치 프로그램은 응용 프로그램을 Applications 디렉토리(예: Mac OS의 /Applications 디렉토리)에 추가합니다.

Linux 컴퓨터에서 AIR는 자동으로 다음을 수행합니다.

- /opt에 응용 프로그램을 설치합니다.
- 응용 프로그램에 대한 바탕 화면 단축키 만들기
- [시작] 메뉴 단축키 만들기
- 시스템 패키지 관리자에서 응용 프로그램에 대한 항목을 추가합니다.

4 원하는 옵션을 선택한 다음 [설치] 버튼을 클릭합니다.

5 설치가 완료되면 [완료]를 클릭합니다.

badge.swf 파일 수정

Flex SDK 및 AIR SDK에서는 badge.swf 파일의 소스 파일을 제공합니다. 이러한 파일은 SDK의 samples/badge 폴더에 포함되어 있습니다.

소스 파일	설명
badge.fla	badge.swf 파일을 컴파일하는 데 사용되는 소스 Flash 파일입니다. badge.fla 파일은 SWF 9 파일(Flash Player에 로드할 수 있는 파일)로 컴파일됩니다.
AIRBadge.as	basdge.fla 파일에 사용되는 기본 클래스를 정의하는 ActionScript 3.0 클래스입니다.

Flash Professional을 사용하여 badge.fla 파일의 시각 인터페이스를 다시 설계할 수 있습니다.

AIRBadge 클래스에 정의되어 있는 AIRBadge() 생성자 함수는 http://airdownload.adobe.com/air/browserapi/air.swf에 있는 air.swf 파일을 로드합니다. air.swf 파일에는 연속 설치 기능을 사용하기 위한 코드가 포함됩니다.

air.swf 파일이 성공적으로 로드되면 AIRBadge 클래스의 onInit() 메서드가 호출됩니다.

```
private function onInit(e:Event):void {
    _air = e.target.content;
    switch (_air.getStatus()) {
        case "installed" :
            root.statusMessage.text = "";
            break;
        case "available" :
            if (_appName && _appName.length > 0) {
                root.statusMessage.htmlText = "<p align='center'><font color='#"
                        + _messageColor + "'>In order to run " + _appName +
                        ", this installer will also set up Adobe® AIR®.</font></p>";
            } else {
                root.statusMessage.htmlText = "<p align='center'><font color='#"
                        + _messageColor + "'>In order to run this application, "
                        + "this installer will also set up Adobe® AIR®.</font></p>";
            }
            break;
        case "unavailable" :
            root.statusMessage.htmlText = "<p align='center'><font color='#"
                        + _messageColor
                        + "'>Adobe® AIR® is not available for your system.</font></p>";
            root.buttonBg_mc.enabled = false;
            break;
    }
}
```

이 코드는 로드된 air.swf 파일의 기본 클래스에 전역 _air 변수를 설정합니다. 이 클래스에는 badge.swf 파일이 연속 설치 기능을 호출하기 위해 액세스하는 다음 공용 메서드가 포함됩니다.

메서드	설명
getStatus()	컴퓨터에 런타임이 설치되어 있는지 또는 런타임을 설치할 수 있는지 여부를 확인합니다. 자세한 내용은 230페이지의 "런타임이 설치되어 있는지 확인"을 참조하십시오. • runtimeVersion - 설치할 응용 프로그램에 필요한 런타임의 버전을 나타내는 문자열(예: "1.0.M6")입니다.
installApplication()	사용자의 컴퓨터에 지정한 응용 프로그램을 설치합니다. 자세한 내용은 231페이지의 "브라우저에서 AIR 응용 프로그램 설치"를 참조하십시오. • url - URL을 정의하는 문자열입니다. 상대 URL 경로가 아닌 절대 URL 경로를 사용해야 합니다. • runtimeVersion - 설치할 응용 프로그램에 필요한 런타임의 버전을 나타내는 문자열(예: "2.5")입니다. • arguments - 설치 시 시작되는 경우 응용 프로그램에 전달할 인수입니다. 응용 프로그램 설명자 파일에서 allowBrowserInvocation 요소가 true로 설정되어 있는 경우 설치 시 응용 프로그램이 시작됩니다. 응용 프로그램 설명자 파일에 대한 자세한 내용은 183페이지의 "AIR 응용 프로그램 설명자 파일"을 참조하십시오. 사용자가 설치 시 시작 기능을 선택하여 브라우저에서 연속 설치의 결과로 응용 프로그램이 시작된 경우 응용 프로그램의 NativeApplication 객체는 인수가 전달된 경우에만 BrowserInvokeEvent 객체를 전달합니다. 응용 프로그램에 전달하는 데이터의 보안 문제를 고려하십시오. 자세한 내용은 232페이지의 "브라우저에서 설치된 AIR 응용 프로그램 시작"을 참조하십시오.

url 및 runtimeVersion의 설정은 컨테이너 HTML 페이지의 FlashVars 설정을 통해 SWF 파일에 전달됩니다.

설치 시 응용 프로그램이 자동으로 시작되는 경우 LocalConnection 통신을 사용하여 badge.swf 파일 호출 시 설치된 응용 프로그램이 이 파일과 통신하도록 만들 수 있습니다. 자세한 내용은 다른 Flash Player 및 AIR 인스턴스와의 통신(ActionScript 개발자용) 또는 Communicating with other Flash Player and AIR instances(HTML 개발자용)을 참조하십시오.

응용 프로그램이 설치되어 있는지 여부를 확인하기 위해 air.swf 파일의 getApplicationVersion() 메서드를 호출할 수도 있습니다. 응용 프로그램 설치 프로세스가 시작되기 전이나 설치가 시작된 후에 이 메서드를 호출할 수 있습니다. 자세한 내용은 230페이지의 "AIR 응용 프로그램이 설치되어 있는지 웹 페이지에서 확인"을 참조하십시오.

air.swf 파일 로드

브라우저의 웹 페이지에서 런타임 및 AIR 응용 프로그램과 상호 작용하기 위해 air.swf 파일의 API를 사용하는 고유 SWF 파일을 만들 수 있습니다. air.swf 파일은 http://airdownload.adobe.com/air/browserapi/air.swf에 있습니다. SWF 파일에서 air.swf API를 참조하려면 SWF 파일과 같은 응용 프로그램 도메인에 air.swf 파일을 로드합니다. 다음 코드에서는 로드하는 SWF 파일의 응용 프로그램 도메인에 air.swf 파일을 로드하는 예제를 보여 줍니다.

```
 var airSWF:Object; // This is the reference to the main class of air.swf
var airSWFLoader:Loader = new Loader(); // Used to load the SWF
var loaderContext:LoaderContext = new LoaderContext();
                                  // Used to set the application domain

loaderContext.applicationDomain = ApplicationDomain.currentDomain;

airSWFLoader.contentLoaderInfo.addEventListener(Event.INIT, onInit);
airSWFLoader.load(new URLRequest("http://airdownload.adobe.com/air/browserapi/air.swf"),
                      loaderContext);

function onInit(e:Event):void
{
    airSWF = e.target.content;
}
```

air.swf 파일이 로드되면(Loader 객체의 contentLoaderInfo 객체가 init 이벤트를 전달하면) 이후 단원에서 설명하는 모든 air.swf API를 호출할 수 있습니다.

참고: AIR SDK 및 Flex SDK와 함께 제공되는 badge.swf 파일은 air.swf 파일을 자동으로 로드합니다. 226페이지의 "badge.swf 파일을 사용하여 AIR 응용 프로그램 설치"를 참조하십시오. 이 단원의 지침은 air.swf 파일을 로드하는 고유 SWF 파일을 만드는 데 적용됩니다.

런타임이 설치되어 있는지 확인

SWF 파일은 http://airdownload.adobe.com/air/browserapi/air.swf에서 로드된 air.swf 파일의 getStatus() 메서드를 호출하여 런타임이 설치되어 있는지 여부를 확인할 수 있습니다. 자세한 내용은 229페이지의 "air.swf 파일 로드"를 참조하십시오.

air.swf 파일이 로드되면 SWF 파일은 air.swf 파일의 getStatus() 메서드를 다음과 같이 호출할 수 있습니다.

```
var status:String = airSWF.getStatus();
```

getStatus() 메서드는 컴퓨터에 있는 런타임의 상태에 따라 다음 문자열 값 중 하나를 반환합니다.

문자열 값	설명
"available"	런타임을 이 컴퓨터에 설치할 수 있지만 현재 설치되어 있지 않습니다.
"unavailable"	이 컴퓨터에 런타임을 설치할 수 없습니다.
"installed"	이 컴퓨터에 런타임이 설치되어 있습니다.

필요한 버전의 Flash Player, 즉 버전 9 업데이트 3 이상(Windows 및 Mac OS) 또는 버전 10(Linux)이 브라우저에 설치되어 있지 않은 경우 getStatus() 메서드에서 오류가 발생합니다.

AIR 응용 프로그램이 설치되어 있는지 웹 페이지에서 확인

SWF 파일은 http://airdownload.adobe.com/air/browserapi/air.swf에서 로드된 air.swf 파일의 getApplicationVersion() 메서드를 호출하여 응용 프로그램 ID와 제작자 ID가 일치하는 AIR 응용 프로그램이 설치되어 있는지 여부를 확인할 수 있습니다. 자세한 내용은 229페이지의 "air.swf 파일 로드"를 참조하십시오.

air.swf 파일이 로드되면 SWF 파일은 air.swf 파일의 getApplicationVersion() 메서드를 다음과 같이 호출할 수 있습니다.

```
var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
airSWF.getApplicationVersion(appID, pubID, versionDetectCallback);

function versionDetectCallback(version:String):void
{
    if (version == null)
    {
        trace("Not installed.");
        // Take appropriate actions. For instance, present the user with
        // an option to install the application.
    }
    else
    {
        trace("Version", version, "installed.");
        // Take appropriate actions. For instance, enable the
        // user interface to launch the application.
    }
}
```

getApplicationVersion() 메서드의 매개 변수는 다음과 같습니다.

매개 변수	설명
appID	응용 프로그램의 응용 프로그램 ID입니다. 자세한 내용은 203페이지의 "id"를 참조하십시오.
pubID	응용 프로그램의 제작자 ID입니다. 자세한 내용은 213페이지의 "publisherID"를 참조하십시오. 해당 응용 프로그램에 제작자 ID가 없는 경우 pubID 매개 변수를 빈 문자열("")로 설정하십시오.
콜백	핸들러 함수의 역할을 하는 콜백 함수입니다. getApplicationVersion() 메서드는 비동기적으로 작동하며 설치된 버전을 검색하거나 설치된 버전이 없음을 확인하면 이 콜백 메서드가 호출됩니다. 콜백 메서드 정의에는 설치된 응용 프로그램의 버전 문자열로 설정된 문자열인 매개 변수가 하나 포함되어야 합니다. 응용 프로그램이 설치되지 않은 경우 이전 코드 샘플에 설명된 대로 null 값이 함수에 전달됩니다.

필요한 버전의 Flash Player, 즉 버전 9 업데이트 3 이상(Windows 및 Mac OS) 또는 버전 10(Linux)이 브라우저에 설치되어 있지 않은 경우 getApplicationVersion() 메서드에서 오류가 발생합니다.

참고: AIR 1.5.3부터는 제작자 ID가 더 이상 사용되지 않습니다. 더 이상 응용 프로그램에 제작자 ID가 자동으로 할당되지 않습니다. 이전 버전과의 호환성을 위해 응용 프로그램에서 제작자 ID를 지정하는 것은 가능합니다.

브라우저에서 AIR 응용 프로그램 설치

SWF 파일은 http://airdownload.adobe.com/air/browserapi/air.swf에서 로드된 air.swf 파일의 installApplication() 메서드를 호출하여 AIR 응용 프로그램을 설치할 수 있습니다. 자세한 내용은 229페이지의 "air.swf 파일 로드"를 참조하십시오.

air.swf 파일이 로드되면 SWF 파일은 air.swf 파일의 installApplication() 메서드를 다음과 같이 호출할 수 있습니다.

```
var url:String = "http://www.example.com/myApplication.air";
var runtimeVersion:String = "1.0";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.installApplication(url, runtimeVersion, arguments);
```

installApplication() 메서드는 사용자의 컴퓨터에 지정한 응용 프로그램을 설치합니다. 이 메서드의 매개 변수는 다음과 같습니다.

매개 변수	설명
url	설치할 AIR 파일의 URL을 정의하는 문자열입니다. 상대 URL 경로가 아닌 절대 URL 경로를 사용해야 합니다.
runtimeVersion	설치할 응용 프로그램에 필요한 런타임의 버전을 나타내는 문자열(예: "1.0")입니다.
arguments	설치 시 시작되는 경우 응용 프로그램에 전달할 인수의 배열입니다. 영숫자 문자만 인수로 인식됩니다. 다른 값을 전달해야 할 경우 인코딩 스킴을 사용하십시오. 응용 프로그램 설명자 파일에서 allowBrowserInvocation 요소가 true로 설정되어 있는 경우 설치 시 응용 프로그램이 시작됩니다. 응용 프로그램 설명자 파일에 대한 자세한 내용은 183페이지의 "AIR 응용 프로그램 설명자 파일"을 참조하십시오. 사용자가 설치 시 시작 기능을 선택하여 브라우저에서 연속 설치의 결과로 응용 프로그램이 시작된 경우 응용 프로그램의 NativeApplication 객체는 인수가 전달된 경우에만 BrowserInvokeEvent 객체를 전달합니다. 자세한 내용은 232페이지의 "브라우저에서 설치된 AIR 응용 프로그램 시작"을 참조하십시오.

마우스 클릭과 같은 사용자 이벤트에 대한 이벤트 핸들러에서 호출될 때만 installApplication() 메서드가 작동할 수 있습니다.

필요한 버전의 Flash Player, 즉 버전 9 업데이트 3 이상(Windows 및 Mac OS) 또는 버전 10(Linux)이 브라우저에 설치되어 있지 않은 경우 installApplication() 메서드에서 오류가 발생합니다.

Mac OS에서는 업데이트된 버전의 응용 프로그램을 설치하려면 사용자에게 응용 프로그램 디렉토리에 설치할 수 있는 적절한 시스템 권한이 필요하며 응용 프로그램이 런타임을 업데이트하는 경우 관리 권한이 필요합니다. Windows에서는 사용자에게 관리 권한이 필요합니다.

응용 프로그램이 이미 설치되어 있는지 여부를 확인하기 위해 air.swf 파일의 getApplicationVersion() 메서드를 호출할 수도 있습니다. 응용 프로그램 설치 프로세스가 시작되기 전이나 설치가 시작된 후에 이 메서드를 호출할 수 있습니다. 자세한 내용은 230페이지의 "AIR 응용 프로그램이 설치되어 있는지 웹 페이지에서 확인"을 참조하십시오. 응용 프로그램이 실행되면 LocalConnection 클래스를 사용하여 브라우저의 SWF 내용과 통신할 수 있습니다. 자세한 내용은 다른 Flash Player 및 AIR 인스턴스와의 통신(ActionScript 개발자용) 또는 Communicating with other Flash Player and AIR instances(HTML 개발자용)을 참조하십시오.

브라우저에서 설치된 AIR 응용 프로그램 시작

브라우저 호출 기능을 사용하여 대상 응용 프로그램이 브라우저에서 시작되게 하려면 해당 응용 프로그램 설명자 파일에 다음 설정이 포함되어야 합니다.

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

응용 프로그램 설명자 파일에 대한 자세한 내용은 183페이지의 "AIR 응용 프로그램 설명자 파일"을 참조하십시오.

브라우저의 SWF 파일은 http://airdownload.adobe.com/air/browserapi/air.swf에서 로드된 air.swf 파일의 launchApplication() 메서드를 호출하여 AIR 응용 프로그램을 시작할 수 있습니다. 자세한 내용은 229페이지의 "air.swf 파일 로드"를 참조하십시오.

air.swf 파일이 로드되면 SWF 파일은 air.swf 파일의 launchApplication() 메서드를 다음과 같이 호출할 수 있습니다.

```
var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.launchApplication(appID, pubID, arguments);
```

launchApplication() 메서드는 사용자 인터페이스 SWF 파일의 응용 프로그램 도메인에 로드되는 air.swf 파일의 최상위 수준에 정의되어 있습니다. 이 메서드를 호출하면 응용 프로그램이 설치되어 있으며 응용 프로그램 설명자 파일의 allowBrowserInvocation 설정을 통해 브라우저 호출이 허용된 경우 AIR가 지정한 응용 프로그램을 시작합니다. 이 메서드의 매개 변수는 다음과 같습니다.

매개 변수	설명
appID	시작할 응용 프로그램의 응용 프로그램 ID입니다. 자세한 내용은 203페이지의 "id"를 참조하십시오.
pubID	시작할 응용 프로그램의 제작자 ID입니다. 자세한 내용은 213페이지의 "publisherID"를 참조하십시오. 해당 응용 프로그램에 제작자 ID가 없는 경우 pubID 매개 변수를 빈 문자열("")로 설정하십시오.
arguments	응용 프로그램에 전달할 인수의 배열입니다. 응용 프로그램의 NativeApplication 객체는 arguments 속성이 이 배열로 설정된 BrowserInvokeEvent 이벤트를 전달합니다. 영숫자 문자만 인수로 인식됩니다. 다른 값을 전달해야 할 경우 인코딩 스킴을 사용하십시오.

마우스 클릭과 같은 사용자 이벤트에 대한 이벤트 핸들러에서 호출될 때만 launchApplication() 메서드가 작동할 수 있습니다.

필요한 버전의 Flash Player, 즉 버전 9 업데이트 3 이상(Windows 및 Mac OS) 또는 버전 10(Linux)이 브라우저에 설치되어 있지 않은 경우 launchApplication() 메서드에서 오류가 발생합니다.

응용 프로그램 설명자 파일에서 allowBrowserInvocation 요소가 false로 설정된 경우 launchApplication() 메서드를 호출해도 효과가 없습니다.

응용 프로그램을 시작하는 사용자 인터페이스를 표시하기 전에 air.swf 파일에서 getApplicationVersion() 메서드를 호출할 수 있습니다. 자세한 내용은 230페이지의 "AIR 응용 프로그램이 설치되어 있는지 웹 페이지에서 확인"을 참조하십시오.

브라우저 호출 기능을 통해 응용 프로그램을 호출하면 응용 프로그램의 NativeApplication 객체가 BrowserInvokeEvent 객체를 전달합니다. 자세한 내용은 브라우저에서 AIR 응용 프로그램 호출(ActionScript 개발자용) 또는 Invoking an AIR application from the browser(HTML 개발자용)을 참조하십시오.

브라우저 호출 기능을 사용하는 경우 보안 영향을 고려해야 합니다. 이러한 보안 영향은 브라우저에서 AIR 응용 프로그램 호출(ActionScript 개발자용) 및 Invoking an AIR application from the browser(HTML 개발자용)에 설명되어 있습니다.

응용 프로그램이 실행되면 LocalConnection 클래스를 사용하여 브라우저의 SWF 내용과 통신할 수 있습니다. 자세한 내용은 다른 Flash Player 및 AIR 인스턴스와의 통신(ActionScript 개발자용) 또는 Communicating with other Flash Player and AIR instances(HTML 개발자용)을 참조하십시오.

참고: AIR 1.5.3부터는 제작자 ID가 더 이상 사용되지 않습니다. 더 이상 응용 프로그램에 제작자 ID가 자동으로 할당되지 않습니다. 이전 버전과의 호환성을 위해 응용 프로그램에서 제작자 ID를 지정하는 것은 가능합니다.

17장: AIR 응용 프로그램 업데이트

사용자는 컴퓨터 또는 브라우저(간편한 설치 기능 사용)에서 AIR 파일을 두 번 클릭하여 AIR 응용 프로그램을 설치하거나 업데이트할 수 있습니다. Adobe® AIR® 설치 응용 프로그램은 설치를 관리하고 사용자가 기존 응용 프로그램을 업데이트하려고 할 경우 경고를 표시합니다.

그러나 Updater 클래스를 사용하여 설치된 응용 프로그램이 새 버전으로 업데이트되게 할 수도 있습니다. 설치된 응용 프로그램은 새 버전을 다운로드하고 설치할 수 있음을 감지할 수 있습니다. Updater 클래스에는 사용자 컴퓨터의 AIR 파일을 가리키고 해당 버전으로 업데이트할 수 있게 하는 update() 메서드가 포함되어 있습니다. Updater 클래스를 사용하려면 응용 프로그램이 AIR 파일로 패키지화되어야 합니다. 기본 실행 파일 또는 패키지로 패키지화된 응용 프로그램은 기본 플랫폼에서 제공하는 업데이트 기능을 사용해야 합니다.

업데이트 AIR 파일의 응용 프로그램 ID와 제작자 ID가 모두 일치해야 응용 프로그램이 업데이트됩니다. 제작자 ID는 서명 인증서에서 파생됩니다. 업데이트와 업데이트할 응용 프로그램이 모두 동일 인증서로 서명되어야 합니다.

AIR 1.5.3 이상의 경우 응용 프로그램 설명자 파일에는 <publisherID> 요소가 포함됩니다. AIR 1.5.2 이하를 사용하여 개발된 응용 프로그램 버전이 있으면 이 요소를 사용해야 합니다. 자세한 내용은 213페이지의 “publisherID”를 참조하십시오.

AIR 1.1 이상부터 응용 프로그램을 마이그레이션하여 새 코드 서명 인증서를 사용할 수 있습니다. 응용 프로그램을 마이그레이션하여 새 서명을 사용하려면 업데이트 AIR 파일을 새 인증서와 원래 인증서를 둘 다 사용하여 서명해야 합니다. 인증서 마이그레이션은 단방향 프로세스입니다. 마이그레이션 후에는 새 인증서(또는 두 인증서)로 서명된 AIR 파일만 기존 설치에 대한 업데이트로 인식됩니다.

응용 프로그램의 업데이트 관리는 복잡할 수 있습니다. AIR 1.5에는 **AdobeAIR 응용 프로그램에 대한 새로운업데이트 프레임워크**가 포함됩니다. 이 프레임워크에서는 개발자가 AIR 응용 프로그램에서 적절한 업데이트 기능을 제공하는 데 도움이 되는 API를 제공합니다.

인증서 마이그레이션을 사용하여 자체 서명된 인증서를 상용 코드 서명 인증서로 변경하거나 자체 서명된 인증서 또는 상용 인증서를 다른 자체 서명된 인증서 또는 상용 인증서로 변경할 수 있습니다. 인증서를 마이그레이션하지 않으면 새 버전을 설치하기 전에 기존 사용자가 현재 응용 프로그램 버전을 제거해야 합니다. 자세한 내용은 174페이지의 “인증서 변경”을 참조하십시오.

응용 프로그램에는 업데이트 메커니즘을 포함하는 것이 좋습니다. 응용 프로그램의 새 버전을 만들 경우 업데이트 메커니즘을 사용하면 사용자에게 새 버전을 설치하도록 알릴 수 있습니다.

AIR 응용 프로그램 설치 프로그램은 AIR 응용 프로그램이 설치, 업데이트 또는 제거될 때 로그 파일을 생성합니다. 이러한 로그를 참조하여 설치 문제의 원인을 확인할 수 있습니다. 설치 로그를 참조하십시오.

참고: 새 버전의 Adobe AIR 런타임에는 업데이트된 WebKit 버전이 포함될 수 있습니다. 업데이트된 WebKit 버전은 배포된 AIR 응용 프로그램의 HTML 내용을 예상치 않게 변경시킬 수 있습니다. 이러한 변경을 위해서는 응용 프로그램을 업데이트해야 할 수 있습니다. 업데이트 메커니즘을 사용하면 응용 프로그램의 새 버전을 사용자에게 알릴 수 있습니다. 자세한 내용은 HTML 환경(ActionScript 개발자용) 또는 About the HTML environment(HTML 개발자용)을 참조하십시오.

응용 프로그램 업데이트

Updater 클래스(flash.desktop 패키지에 포함됨)에 있는 update() 메서드는 현재 실행되는 응용 프로그램을 다른 버전으로 업데이트하는 데 사용할 수 있습니다. 예를 들어, 데스크톱에 특정 버전의 AIR 파일("Sample_App_v2.air")이 있는 경우 다음 코드는 응용 프로그램을 업데이트합니다.

ActionScript 예제:

```
var updater:Updater = new Updater();
var airFile:File = File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version:String = "2.01";
updater.update(airFile, version);
```

JavaScript 예제:

```
var updater = new air.Updater();
var airFile = air.File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version = "2.01";
updater.update(airFile, version);
```

응용 프로그램에서 Updater 클래스를 사용하기 전에 사용자나 응용 프로그램은 업데이트된 버전의 AIR 파일을 컴퓨터에 다운로드해야 합니다. 자세한 내용은 237페이지의 “사용자 컴퓨터에 AIR 파일 다운로드”를 참조하십시오.

Updater.update() 메서드 호출의 결과

런타임의 응용 프로그램에서 update() 메서드를 호출하면 런타임은 응용 프로그램을 닫은 다음 AIR 파일에서 새 버전을 설치하려고 합니다. 런타임은 AIR 파일에 지정된 응용 프로그램 ID 및 제작자 ID가 update() 메서드를 호출하는 응용 프로그램의 응용 프로그램 ID 및 제작자 ID와 일치하는지 확인합니다. 응용 프로그램 ID 및 제작자 ID에 대한 자세한 내용은 183페이지의 “AIR 응용 프로그램 설명자 파일”을 참조하십시오. 또한 버전 문자열이 update() 메서드에 전달된 version 문자열과 일치하는지도 확인합니다. 런타임은 설치가 성공적으로 완료되면 새 버전의 응용 프로그램을 열고, 그렇지 않으면(설치가 완료될 수 없으면) 기존(설치 전) 버전의 응용 프로그램을 다시 엽니다.

Mac OS에서는 업데이트된 버전의 응용 프로그램을 설치하려면 사용자에게 응용 프로그램 디렉토리에 설치할 수 있는 적절한 시스템 권한이 필요합니다. Windows 및 Linux에서는 사용자에게 관리 권한이 있어야 합니다.

업데이트된 버전의 응용 프로그램에 업데이트된 버전의 런타임이 필요한 경우에는 새 런타임 버전이 설치됩니다. 런타임을 업데이트하려면 사용자에게 컴퓨터에 대한 관리 권한이 필요합니다.

ADL을 사용하여 응용 프로그램을 테스트할 때 update() 메서드를 호출하면 런타임 예외가 발생합니다.

버전 문자열

update() 메서드의 version 매개 변수로 지정된 문자열이 응용 프로그램 설명자 파일의 version 또는 versionNumber 요소에 있는 문자열과 일치해야 AIR 파일이 설치됩니다. version 매개 변수를 지정하는 것은 보안상의 이유로 필요합니다. 응용 프로그램에서 AIR 파일의 버전 번호를 확인하도록 설정하면 응용 프로그램에서 의도하지 않게 이전 버전을 설치하는 일은 없어집니다. 이전 버전의 응용 프로그램에는 현재 설치된 응용 프로그램에서 수정된 보안 취약점이 있을 수 있습니다. 또한 응용 프로그램에서는 AIR 파일의 버전 문자열을 설치된 응용 프로그램의 버전 문자열과 비교하여 다운그레이드 공격을 방지해야 합니다.

AIR 2.5 이전 버전에서는 포맷에 관계없이 모든 버전 문자열을 사용할 수 있습니다. 예를 들어 "2.01" 또는 "version 2"일 수 있습니다. AIR 2.5 이상에서는 버전 문자열이 마침표로 분리된 최대 3개의 세 자리 숫자여야 합니다. 예를 들어 “.0”, “1.0” 및 “67.89.999”가 모두 유효한 버전 번호입니다. 응용 프로그램을 업데이트하기 전에 업데이트 버전 문자열의 유효성을 검사해야 합니다.

Adobe AIR 응용 프로그램에서 AIR 파일을 웹을 통해 다운로드하는 경우 웹 서비스에서 Adobe AIR 응용 프로그램에 다운로드할 버전을 알릴 수 있는 메커니즘을 마련하는 것이 좋습니다. 이렇게 하면 응용 프로그램에서 이 문자열을 update() 메서드의 version 매개 변수로 사용할 수 있습니다. AIR 파일의 버전을 알 수 없는 다른 방법으로 AIR 파일을 얻은 경우 AIR 응용 프로그램에서 AIR 파일을 검사하여 버전 정보를 확인할 수 있습니다. AIR 파일은 ZIP 압축 파일이고 응용 프로그램 설명자 파일은 압축 파일의 두 번째 레코드입니다.

응용 프로그램 설명자 파일에 대한 자세한 내용은 183페이지의 “AIR 응용 프로그램 설명자 파일”을 참조하십시오.

응용 프로그램 업데이트를 위한 서명 작업 과정

애드혹 방식으로 업데이트를 제작하면 여러 응용 프로그램 버전을 관리하는 작업이 복잡해지고 인증서 만료 날짜를 추적하는 일도 어려워집니다. 업데이트를 제작하기 전에 인증서가 만료될 수도 있습니다.

Adobe AIR 런타임은 마이그레이션 서명 없이 제작된 응용 프로그램 업데이트를 새로운 응용 프로그램으로 취급합니다. 사용자는 먼저 현재 AIR 응용 프로그램을 제거한 후에야 응용 프로그램 업데이트를 설치할 수 있습니다.

문제를 해결하려면 별도의 배포 URL에 각각의 업데이트된 응용 프로그램을 최신 인증서와 함께 업로드하십시오. 인증서가 180일의 유예 기간 안에 있을 때는 마이그레이션 서명을 적용할 것을 알리는 메커니즘을 포함하십시오. 자세한 내용은 179페이지의 “업데이트된 버전의 AIR 응용 프로그램에 서명”을 참조하십시오.

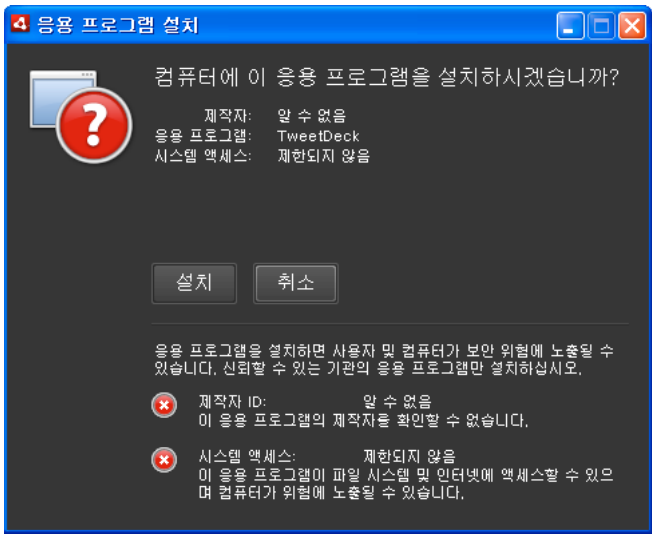
서명을 적용하는 방법에 대한 자세한 내용은 149페이지의 “ADT 명령”을 참조하십시오.

다음 작업을 수행하여 마이그레이션 서명을 적용하는 프로세스를 간소화합니다.

- 각각의 업데이트된 응용 프로그램을 별도의 배포 URL에 업로드합니다.
- 업그레이드 설명자 XML 파일과 업데이트의 최신 인증서를 같은 URL에 업로드합니다.
- 최신 인증서를 사용하여 업데이트된 응용 프로그램에 서명합니다.
- 다른 URL에 있는 이전 버전에 서명할 때 사용한 인증서를 사용하여 업데이트된 응용 프로그램에 마이그레이션 서명을 적용합니다.

사용자 정의 응용 프로그램 업데이트 사용자 인터페이스 제공

AIR에는 기본 업데이트 인터페이스가 포함되어 있습니다.



이 인터페이스는 사용자가 컴퓨터에 응용 프로그램의 버전을 처음 설치할 때 항상 사용됩니다. 그러나 이후 인스턴스의 경우 사용자가 고유의 인터페이스를 정의할 수 있습니다. 응용 프로그램에서 사용자 정의 업데이트 인터페이스를 정의하는 경우 현재 설치된 응용 프로그램에 대한 응용 프로그램 설명자 파일에서 customUpdateUI 요소를 지정합니다.

```
<customUpdateUI>true</customUpdateUI>
```

응용 프로그램이 설치되고 사용자가 설치된 응용 프로그램과 일치하는 응용 프로그램 ID 및 제작자 ID가 포함된 AIR 파일을 열면 런타임은 기본 AIR 응용 프로그램 설치 프로그램이 아니라 응용 프로그램을 엽니다. 자세한 내용은 194페이지의 “customUpdateUI”를 참조하십시오.

응용 프로그램에서 실행된 시기(NativeApplication.nativeApplication 객체가 load 이벤트를 전달한 시기)와 응용 프로그램을 업데이트할지 여부(Updater 클래스 사용)를 확인할 수 있습니다. 응용 프로그램에서 업데이트하도록 결정하는 경우 표준 실행 인터페이스와 다른 고유의 설치 인터페이스를 사용자에게 제공할 수 있습니다.

사용자 컴퓨터에 AIR 파일 다운로드

Updater 클래스를 사용하려면 먼저 사용자나 응용 프로그램이 AIR 파일을 사용자 컴퓨터에 로컬로 저장해야 합니다.

참고: AIR 1.5에는 개발자가 AIR 응용 프로그램에서 적절한 업데이트 기능을 제공하는 데 도움이 되는 업데이트 프레임워크가 포함되어 있습니다. Update 클래스의 update() 메서드를 직접 사용하는 것보다 이 프레임워크를 사용하는 것이 훨씬 쉽습니다. 자세한 내용은 240페이지의 “업데이트 프레임워크 사용”을 참조하십시오.

다음 코드는 AIR 파일을 URL(http://example.com/air/updates/Sample_App_v2.air)에서 읽고 AIR 파일을 응용 프로그램 저장소 디렉토리에 저장합니다.

ActionScript 예제:

```
var urlString:String = "http://example.com/air/updates/Sample_App_v2.air";
var urlReq:URLRequest = new URLRequest(urlString);
var urlStream:URLStream = new URLStream();
var fileData:ByteArray = new ByteArray();
urlStream.addEventListener(Event.COMPLETE, loaded);
urlStream.load(urlReq);

function loaded(event:Event):void {
    urlStream.readBytes(fileData, 0, urlStream.bytesAvailable);
    writeAirFile();
}

function writeAirFile():void {
    var file:File = File.applicationStorageDirectory.resolvePath("My App v2.air");
    var fileStream:FileStream = new FileStream();
    fileStream.open(file, FileMode.WRITE);
    fileStream.writeBytes(fileData, 0, fileData.length);
    fileStream.close();
    trace("The AIR file is written.");
}
```

JavaScript 예제:

```
var urlString = "http://example.com/air/updates/Sample_App_v2.air";
var urlReq = new air.URLRequest(urlString);
var urlStream = new air.URLStream();
var fileData = new air.ByteArray();
urlStream.addEventListener(air.Event.COMPLETE, loaded);
urlStream.load(urlReq);

function loaded(event) {
    urlStream.readBytes(fileData, 0, urlStream.bytesAvailable);
    writeAirFile();
}

function writeAirFile() {
    var file = air.File.desktopDirectory.resolvePath("My App v2.air");
    var fileStream = new air.FileStream();
    fileStream.open(file, air.FileMode.WRITE);
    fileStream.writeBytes(fileData, 0, fileData.length);
    fileStream.close();
    trace("The AIR file is written.");
}
```

자세한 내용은 다음 항목을 참조하십시오.

- 파일 읽기 및 쓰기 작업 과정(ActionScript 개발자용)
- Workflow for reading and writing files(HTML 개발자용)

응용 프로그램이 처음 실행되는지 확인

응용 프로그램을 업데이트한 후 사용자에게 "시작" 또는 "환영" 메시지를 제공할 수 있습니다. 응용 프로그램은 시작될 때 처음 실행되는지 여부를 확인하므로 이러한 메시지를 표시할지 여부를 결정할 수 있습니다.

참고: AIR 1.5에는 개발자가 AIR 응용 프로그램에서 적절한 업데이트 기능을 제공하는 데 도움이 되는 업데이트 프레임워크가 포함되어 있습니다. 이 프레임워크에서는 응용 프로그램 버전이 처음으로 실행되는지 여부를 확인하는 간편한 방법을 제공합니다. 자세한 내용은 240페이지의 “업데이트 프레임워크 사용”을 참조하십시오.

이렇게 하는 한 가지 방법은 응용 프로그램을 초기화할 때 응용 프로그램 저장소 디렉토리에 파일을 저장하는 것입니다. 응용 프로그램은 시작될 때마다 해당 파일이 있는지 확인해야 합니다. 파일이 없으면 현재 사용자의 경우 응용 프로그램이 처음 실행되는 것이고, 파일이 있으면 응용 프로그램이 이미 한 번 이상 실행된 것입니다. 파일이 있고 현재 버전 번호보다 이전의 버전 번호를 포함하는 경우 사용자가 새 버전을 처음 실행하는 것입니다.

다음 Flex 예제에서는 이 개념을 보여 줍니다.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical"
    title="Sample Version Checker Application"
    applicationComplete="system extension()">
    <mx:Script>
        <![CDATA[
            import flash.filesystem.*;
            public var file:File;
            public var currentVersion:String = "1.2";
            public function system extension():void {
                file = File.applicationStorageDirectory;
                file = file.resolvePath("Preferences/version.txt");
                trace(file.nativePath);
                if(file.exists) {
                    checkVersion();
                } else {
                    firstRun();
                }
            }
            private function checkVersion():void {
                var stream:FileStream = new FileStream();
                stream.open(file, FileMode.READ);
                var reversion:String = stream.readUTFBytes(stream.bytesAvailable);
                stream.close();
                if (reversion != currentVersion) {
                    log.text = "You have updated to version " + currentVersion + ".\n";
```

```
                } else {
                    saveFile();
                }
                log.text += "Welcome to the application.";
            }
            private function firstRun():void {
                log.text = "Thank you for installing the application. \n"
                    + "This is the first time you have run it.";
                saveFile();
            }
            private function saveFile():void {
                var stream:FileStream = new FileStream();
                stream.open(file, FileMode.WRITE);
                stream.writeUTFBytes(currentVersion);
                stream.close();
            }
        ]]>
    </mx:Script>
    <mx:TextArea ID="log" width="100%" height="100%" />
</mx:WindowedApplication>
```

다음 예제에서는 JavaScript에서의 이 개념을 보여 줍니다.

```
<html>
    <head>
        <script src="AIRAliases.js" />
        <script>
            var file;
            var currentVersion = "1.2";
            function system extension() {
                file = air.File.appStorageDirectory.resolvePath("Preferences/version.txt");
                air.trace(file.nativePath);
                if(file.exists) {
                    checkVersion();
                } else {
                    firstRun();
                }
            }
            function checkVersion() {
                var stream = new air.FileStream();
                stream.open(file, air.FileMode.READ);
                var reversion = stream.readUTFBytes(stream.bytesAvailable);
                stream.close();
                if (reversion != currentVersion) {
                    window.document.getElementById("log").innerHTML
                            = "You have updated to version " + currentVersion + ".\n";
                } else {
                    saveFile();
                }
                window.document.getElementById("log").innerHTML
                                += "Welcome to the application.";
```

```
            }
            function firstRun() {
                window.document.getElementById("log").innerHTML
                            = "Thank you for installing the application. \n"
                            + "This is the first time you have run it.";
                saveFile();
            }
            function saveFile() {
                var stream = new air.FileStream();
                stream.open(file, air.FileMode.WRITE);
                stream.writeUTFBytes(currentVersion);
                stream.close();
            }
        </script>
    </head>
    <body onLoad="system extension()">
        <textarea ID="log" rows="100%" cols="100%" />
    </body>
</html>
```

응용 프로그램에서 데이터를 응용 프로그램 저장소 디렉토리 등에 로컬로 저장하는 경우 처음 실행될 때 이전 버전에서 이전에 저장된 데이터를 확인할 수 있습니다.

업데이트 프레임워크 사용

응용 프로그램에 대한 업데이트를 관리하는 작업은 번거로울 수 있습니다. **AdobeAIR 응용 프로그램을 위한 업데이트 프레임워크**는 개발자가 AIR 응용 프로그램에서 강력한 업데이트 기능을 구현할 수 있도록 하는 API를 제공합니다. AIR 업데이트 프레임워크는 개발자를 위해 다음 작업을 수행합니다.

- 일정 간격으로 또는 사용자가 요청할 때 업데이트를 주기적으로 확인
- 웹 소스에서 AIR 파일(업데이트) 다운로드
- 새로 설치된 버전의 첫 실행을 사용자에게 알림
- 사용자에게 업데이트를 확인할지 여부 묻기
- 사용자에게 새 업데이트 버전에 대한 정보 표시
- 다운로드 진행률 및 오류 정보를 사용자에게 표시

AIR 업데이트 프레임워크는 응용 프로그램을 위한 샘플 사용자 인터페이스를 제공하며, 응용 프로그램 업데이트에 대한 기본 정보 및 구성 옵션을 사용자에게 제공합니다. 응용 프로그램에서 업데이트 프레임워크에 사용할 사용자 정의 사용자 인터페이스를 정의할 수도 있습니다.

AIR 업데이트 프레임워크에서는 AIR 응용 프로그램의 업데이트 버전에 대한 정보를 간단한 XML 구성 파일에 저장할 수 있습니다. 대부분의 응용 프로그램에서는 일부 기본 코드가 포함되도록 이러한 구성 파일을 설정하면 최종 사용자가 쉽게 업데이트 작업을 수행할 수 있습니다.

Adobe AIR에는 업데이트 프레임워크 외에도 AIR 응용 프로그램이 새 버전으로 업그레이드하는 데 사용할 수 있는 Updater 클래스가 포함되어 있습니다. Updater 클래스를 사용하여 응용 프로그램을 사용자 컴퓨터의 AIR 파일에 포함된 버전으로 업그레이드할 수 있습니다. 그러나 업그레이드 관리에는 단순히 응용 프로그램이 로컬에 저장한 AIR 파일을 기반으로 업데이트되는 것 이상의 작업이 포함될 수 있습니다.

AIR 업데이트 프레임워크 파일

AIR 업데이트 프레임워크는 AIR 2 SDK의 frameworks/libs/air 디렉토리에 있으며 다음 파일을 포함합니다.

- applicationupdater.swc - ActionScript에서 사용되는 업데이트 라이브러리의 기본적인 기능을 정의합니다. 이 버전은 사용자 인터페이스를 포함하지 않습니다.
- applicationupdater.swf - JavaScript에서 사용되는 업데이트 라이브러리의 기본적인 기능을 정의합니다. 이 버전은 사용자 인터페이스를 포함하지 않습니다.
- applicationupdater_ui.swc - 응용 프로그램이 업데이트 옵션을 표시하는 데 사용할 수 있는 사용자 인터페이스를 포함하여 Flex 4 버전 업데이트 라이브러리의 기본적인 기능을 정의합니다.
- applicationupdater_ui.swf - 응용 프로그램이 업데이트 옵션을 표시하는 데 사용할 수 있는 사용자 인터페이스를 포함하여 JavaScript 버전 업데이트 라이브러리의 기본적인 기능을 정의합니다.

자세한 내용은 다음 단원을 참조하십시오.

- 241페이지의 “Flex 개발 환경 설정”
- 241페이지의 “HTML 기반 AIR 응용 프로그램에 프레임워크 파일 포함”
- 242페이지의 “기본 예제: ApplicationUpdaterUI 버전 사용”

Flex 개발 환경 설정

AIR 2 SDK의 frameworks/libs/air 디렉토리에 있는 SWC 파일은 Flex 및 Flash 개발에 사용할 수 있는 클래스를 정의합니다.

Flex SDK를 사용하여 컴파일할 때 업데이트 프레임워크를 사용하려면 amxmlc 컴파일러에 대한 호출에 ApplicationUpdater.swc 또는 ApplicationUpdater_UI.swc 파일을 포함합니다. 다음 예제에서는 컴파일러가 Flex SDK 디렉토리의 lib 하위 디렉토리에 있는 ApplicationUpdater.swc 파일을 로드합니다.

```
amxmlc -library-path+=lib/ApplicationUpdater.swc  -- myApp.mxml
```

다음 예제에서는 컴파일러가 Flex SDK 디렉토리의 lib 하위 디렉토리에 있는 ApplicationUpdater_UI.swc 파일을 로드합니다.

```
amxmlc -library-path+=lib/ApplicationUpdater_UI.swc  -- myApp.mxml
```

Flash Builder를 사용하여 개발할 경우 [Properties] 대화 상자에서 [Flex Build Path settings]의 [Library Path] 탭에 SWC 파일을 추가합니다.

SWC 파일을 amxmlc 컴파일러(Flex SDK 사용) 또는 Flash Builder에서 참조할 디렉토리로 복사해야 합니다.

HTML 기반 AIR 응용 프로그램에 프레임워크 파일 포함

업데이트 프레임워크의 frameworks/html 디렉토리에는 다음 SWF 파일이 포함되어 있습니다.

- applicationupdater.swf - 사용자 인터페이스 없이 업데이트 라이브러리의 기본 기능을 정의합니다.
- applicationupdater_ui.swf - 응용 프로그램이 업데이트 옵션을 표시하는 데 사용할 수 있는 사용자 인터페이스를 포함하여 업데이트 라이브러리의 기본 기능을 정의합니다.

AIR 응용 프로그램의 JavaScript 코드에서는 SWF 파일에 정의된 클래스를 사용할 수 있습니다.

업데이트 프레임워크를 사용하려면 applicationupdater.swf 또는 applicationupdater_ui.swf 파일을 응용 프로그램 디렉토리 또는 하위 디렉토리에 포함합니다. 그런 다음 JavaScript 코드에서 프레임워크를 사용할 HTML 파일에 파일을 로드하는 script 태그를 포함합니다.

```
<script src="applicationUpdater.swf" type="application/x-shockwave-flash"/>
```

또는 이 script 태그를 사용하여 applicationupdater_ui.swf 파일을 로드합니다.

```
<script src="applicationupdater_ui.swf" type="application/x-shockwave-flash"/>
```

이러한 두 파일에서 정의되는 API는 이 문서의 나머지 부분에서 설명합니다.

기본 예제: ApplicationUpdaterUI 버전 사용

업데이트 프레임워크의 ApplicationUpdaterUI 버전은 응용 프로그램에서 쉽게 사용할 수 있는 기본 인터페이스를 제공합니다. 다음은 기본 예제입니다.

먼저 업데이트 프레임워크를 호출하는 AIR 응용 프로그램을 만듭니다.

1 응용 프로그램이 HTML 기반 AIR 응용 프로그램인 경우 applicationupdaterui.swf 파일을 로드합니다.

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

2 AIR 응용 프로그램 논리에서 ApplicationUpdaterUI 객체를 인스턴스화합니다.

ActionScript에서는 다음 코드를 사용합니다.

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

JavaScript에서는 다음 코드를 사용합니다.

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

응용 프로그램이 로드될 때 실행되는 초기화 기능에 이 코드를 추가할 수 있습니다.

3 updateConfig.xml이라는 텍스트 파일을 만들어 다음 내용을 추가합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://ns.adobe.com/air/framework/update/configuration/1.0">
    <url>http://example.com/updates/update.xml</url>
    <delay>1</delay>
</configuration>
```

updateConfig.xml 파일의 URL 요소가 웹 서버에 있는 업데이트 설명자 파일의 최종 위치와 일치하도록 편집합니다(다음 절차 참조).

delay는 응용 프로그램이 다음 업데이트 확인 때까지 대기하는 기간(일)입니다.

4 updateConfig.xml 파일을 AIR 응용 프로그램의 project 디렉토리에 추가합니다.

5 업데이트 프로그램 객체가 updateConfig.xml 파일을 참조하게 하도록 객체의 initialize() 메서드를 호출합니다.

ActionScript에서는 다음 코드를 사용합니다.

```
appUpdater.configurationFile = new File("app:/updateConfig.xml");
appUpdater.initialize();
```

JavaScript에서는 다음 코드를 사용합니다.

```
appUpdater.configurationFile = new air.File("app:/updateConfig.xml");
appUpdater.initialize();
```

6 첫 번째 응용 프로그램과 버전이 다른 두 번째 AIR 응용 프로그램 버전을 만듭니다. 버전은 응용 프로그램 설명자 파일의 version 요소에 지정됩니다.

그런 다음 AIR 응용 프로그램의 업데이트 버전을 웹 서버에 추가합니다.

1 AIR 파일의 업데이트 버전을 웹 서버에 배치합니다.

2 updateDescriptor.2.5.xml이라는 텍스트 파일을 만들어 다음 내용을 추가합니다.

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/2.5">
       <versionNumber>1.1</versionNumber>
       <url>http://example.com/updates/sample_1.1.air</url>
       <description>This is the latest version of the Sample application.</description>
    </update>
```

updateDescriptor.xml 파일의 versionNumber, URL 및 description을 업데이트 AIR 파일과 일치하도록 편집합니다. 이 업데이트 설명자 포맷은 AIR 2.5 SDK 이상에 포함된 업데이트 프레임워크를 사용하는 응용 프로그램에 의해 사용됩니다.

3 updateDescriptor.1.0.xml이라는 텍스트 파일을 만들어 다음 내용을 추가합니다.

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1</version>
       <url>http://example.com/updates/sample_1.1.air</url>
       <description>This is the latest version of the Sample application.</description>
    </update>
```

updateDescriptor.xml 파일의 version, URL 및 description을 업데이트 AIR 파일과 일치하도록 편집합니다. 이 업데이트 설명자 포맷은 AIR 2 SDK 이하에 포함된 업데이트 프레임워크를 사용하는 응용 프로그램에 의해 사용됩니다.

참고: 이 두 번째 업데이트 설명자 파일을 만드는 작업은 AIR 2.5 이전에 만든 응용 프로그램에 대해 업데이트를 지원할 때만 수행하면 됩니다.

4 updateDescriptor.2.5.xml 및 updateDescriptor.1.0.xml 파일을 업데이트 AIR 파일이 포함되어 있는 웹 서버 디렉토리에 추가합니다.

이것은 기본적인 예제이지만 많은 응용 프로그램에 충분한 업데이트 기능을 제공합니다. 이 문서의 나머지 부분에서는 업데이트 프레임워크를 사용자 요구에 가장 잘 맞게 사용하는 방법을 설명합니다.

업데이트 프레임워크 사용과 관련된 다른 예제를 보려면 Adobe AIR 개발자 센터의 다음 샘플 응용 프로그램을 참조하십시오.

- Flash 기반 응용 프로그램의 업데이트 프레임워크
(http://www.adobe.com/go/learn_air_qs_update_framework_flash_kr)

AIR 2.5로 업데이트

AIR 2.5에서 응용 프로그램에 버전 번호를 할당하는 규칙이 변경되었기 때문에 AIR 2 업데이트 프레임워크는 AIR 2.5 응용 프로그램 설명자에 있는 버전 정보를 파싱할 수 없습니다. 이러한 비호환성 때문에 AIR 2.5 SDK를 사용하도록 응용 프로그램을 업데이트하기 전에 새 업데이트 프레임워크를 사용하도록 응용 프로그램을 업데이트해야 합니다. 따라서 AIR 2.5 이전 버전에서 AIR 2.5 이상으로 응용 프로그램을 업데이트할 때는 두 개의 업데이트가 필요합니다. 첫 번째 업데이트는 AIR 2 네임스페이스를 사용하고 AIR 2.5 업데이트 프레임워크 라이브러리를 포함해야 합니다(여전히 AIR 2.5 SDK를 사용하여 응용 프로그램 패키지를 만들 수 있음). 두 번째 업데이트는 AIR 2.5 네임스페이스를 사용하고 응용 프로그램의 새 기능을 포함할 수 있습니다.

AIR Updater 클래스를 직접 사용하여 중간 업데이트가 AIR 2.5 응용 프로그램으로 업데이트되는 것을 제외하고는 아무것도 수행되지 않도록 할 수도 있습니다.

다음 예제에서는 버전 1.0에서 2.0으로 응용 프로그램을 업데이트하는 방법을 보여 줍니다. 버전 1.0은 이전 2.0 네임스페이스를 사용합니다. 버전 2.0은 2.5 네임스페이스를 사용하며, AIR 2.5 API를 사용하여 구현된 새로운 기능을 포함하고 있습니다.

1 응용 프로그램의 버전 1.0을 기준으로 응용 프로그램의 중간 버전인 버전 1.0.1을 만듭니다.

a 응용 프로그램을 만드는 동안 AIR 2.5 Application Updater 프레임워크를 사용합니다.

참고: Flash 기술을 기반으로 하는 AIR 응용 프로그램에는 applicationupdater.swc 또는 applicationupdater_ui.swc를 사용하고, HTML 기반 AIR 응용 프로그램에는 applicationupdater.swf 또는 applicationupdater_ui.swf를 사용합니다.

b 이전 네임스페이스 및 아래에 표시된 버전을 사용하여 버전 1.0.1에 대한 업데이트 설명자 파일을 만듭니다.

```
<?xml version="1.0" encoding="utf-8"?>
    <update xmlns="http://ns.adobe.com/air/framework/update/description/2.0">
        <version>1.0.1</version>
        <url>http://example.com/updates/sample_1.0.1.air</url>
        <description>This is the intermediate version.</description>
    </update>
```

2 AIR 2.5 API 및 2.5 네임스페이스를 사용하는 응용 프로그램의 버전 2.0을 만듭니다.

3 업데이트 설명자를 만들어 응용 프로그램을 1.0.1 버전에서 2.0 버전으로 업데이트합니다.

```
<?xml version="1.0" encoding="utf-8"?>
    <update xmlns="http://ns.adobe.com/air/framework/update/description/2.5">
        <version>2.0</version>
        <url>http://example.com/updates/sample_2.0.air</url>
        <description>This is the intermediate version.</description>
    </update>
```

업데이트 설명자 파일 정의 및 웹 서버에 AIR 파일 추가

AIR 업데이트 프레임워크를 사용할 때는 웹 서버에 저장된 업데이트 설명자 파일에 사용 가능한 업데이트에 대한 기본 정보를 정의합니다. 업데이트 설명자 파일은 간단한 XML 파일입니다. 응용 프로그램에 포함된 업데이트 프레임워크는 이 파일을 확인하여 새 버전이 업로드되었는지 확인합니다.

업데이트 설명자 파일의 포맷은 AIR 2.5에서 변경되었습니다. 새로운 포맷은 다른 네임스페이스를 사용합니다. 원래의 네임스페이스는 “http://ns.adobe.com/air/framework/update/description/1.0”입니다. AIR 2.5 네임스페이스는 “http://ns.adobe.com/air/framework/update/description/2.5”입니다.

AIR 2.5 이전에 생성된 AIR 응용 프로그램은 버전 1.0 업데이트 설명자만 읽을 수 있습니다. AIR 2.5 이상에 포함된 업데이터 프레임워크를 사용하여 생성된 AIR 응용 프로그램은 버전 2.5 업데이트 설명자만 읽을 수 있습니다. 이러한 버전 비호환성 때문에 종종 두 개의 업데이트 설명자 파일을 만들어야 합니다. AIR 2.5 버전의 응용 프로그램에 있는 업데이트 논리는 새 포맷을 사용하는 업데이트 설명자를 다운로드해야 합니다. 이전 버전의 AIR 응용 프로그램은 계속 원래의 포맷을 사용해야 합니다. AIR 2.5 이전에 만든 버전에 대한 지원을 중지하기 전까지 업데이트를 릴리스할 때마다 두 파일을 모두 수정해야 합니다.

업데이트 설명자 파일에는 다음 데이터가 포함됩니다.

- versionNumber - AIR 응용 프로그램의 새 버전입니다. AIR 2.5 응용 프로그램을 업데이트하는 데 사용된 업데이트 설명자에서 versionNumber 요소를 사용합니다. 이 값은 새 AIR 응용 프로그램 설명자 파일의 versionNumber 요소에 사용된 것과 동일한 문자열이어야 합니다. 업데이트 설명자 파일의 버전 번호가 업데이트 AIR 파일의 버전 번호와 일치하지 않으면 업데이트 프레임워크에서 예외가 발생합니다.
- version - AIR 응용 프로그램의 새 버전입니다. AIR 2.5 이전에 만든 응용 프로그램을 업데이트하는 데 사용된 업데이트 설명자에서 version 요소를 사용합니다. 이 값은 새 AIR 응용 프로그램 설명자 파일의 version 요소에 사용된 것과 동일한 문자열이어야 합니다. 업데이트 설명자 파일의 버전이 업데이트 AIR 파일의 버전과 일치하지 않으면 업데이트 프레임워크에서 예외가 발생합니다.
- versionLabel - 사용자에게 표시하기 위한 목적의, 사람이 읽을 수 있는 버전 문자열입니다. versionLabel은 선택 사항이지만, 버전 2.5 업데이트 설명자 파일에서만 지정될 수 있습니다. 응용 프로그램 설명자에서 versionLabel을 사용하는 경우에만 사용하고 같은 값으로 설정하십시오.
- url - 업데이트 AIR 파일의 위치입니다. 이 파일은 AIR 응용 프로그램의 업데이트 버전을 포함합니다.
- description - 새 버전과 관련된 세부 사항입니다. 이 정보는 업데이트 프로세스 중에 사용자에게 표시될 수 있습니다.

version 및 url 요소는 필수이고, description 요소는 선택 사항입니다.

다음은 샘플 버전 2.5 업데이트 설명자 파일입니다.

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/2.5">
       <versionNumber>1.1.1</versionNumber>
       <url>http://example.com/updates/sample_1.1.1.air</url>
       <description>This is the latest version of the Sample application.</description>
    </update>
```

그리고 다음은 샘플 버전 1.0 업데이트 설명자 파일입니다.

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1.1</version>
       <url>http://example.com/updates/sample_1.1.1.air</url>
       <description>This is the latest version of the Sample application.</description>
    </update>
```

여러 언어를 사용하여 description 태그를 정의하려면 lang 특성을 정의하는 여러 text 요소를 사용합니다.

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/2.5">
       <versionNumber>1.1.1</versionNumber>
       <url>http://example.com/updates/sample_1.1.1.air</url>
       <description>
           <text xml:lang="en">English description</text>
           <text xml:lang="fr">French description</text>
           <text xml:lang="ro">Romanian description</text>
       </description>
    </update>
```

업데이트 설명자 파일을 업데이트 AIR 파일과 함께 웹 서버에 배치합니다.

업데이트 설명자에 포함된 templates 디렉토리에는 샘플 업데이트 설명자 파일이 포함됩니다. 여기에는 단일 언어 버전과 여러 언어 버전이 모두 포함됩니다.

업데이트 프로그램 객체 인스턴스화

코드에서 AIR 업데이트 프레임워크를 로드한 후(241페이지의 “Flex 개발 환경 설정” 및 241페이지의 “HTML 기반 AIR 응용 프로그램에 프레임워크 파일 포함” 참조) 다음과 같이 업데이트 프로그램 객체를 인스턴스화해야 합니다

ActionScript 예제:

```
var appUpdater:ApplicationUpdater = new ApplicationUpdater();
```

JavaScript 예제:

```
var appUpdater = new runtime.air.update.ApplicationUpdater();
```

앞의 코드에서는 사용자 인터페이스를 제공하지 않는 ApplicationUpdater 클래스를 사용합니다. 사용자 인터페이스를 제공하는 ApplicationUpdaterUI 클래스를 사용하려면 다음을 사용합니다.

ActionScript 예제:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

JavaScript 예제:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

이 문서의 나머지 코드 샘플에서는 appUpdater라는 업데이트 프로그램 객체를 인스턴스화한 것으로 가정합니다.

업데이트 설정 구성

ApplicationUpdater 및 ApplicationUpdaterUI는 모두 응용 프로그램과 함께 제공되는 구성 파일을 통해 또는 응용 프로그램의 ActionScript나 JavaScript를 통해 구성할 수 있습니다.

XML 구성 파일에서 업데이트 설정 정의

업데이트 구성 파일은 XML 파일이며, 다음 요소를 포함할 수 있습니다.

- updateURL - 문자열입니다. 원격 서버에 있는 업데이트 설명자의 위치를 나타냅니다. 유효한 URLRequest 위치를 사용할 수 있습니다. 구성 파일이나 스크립트를 통해 updateURL 속성을 정의해야 합니다(244페이지의 “업데이트 설명자 파일 정의

및 웹 서버에 AIR 파일 추가" 참조). 업데이트 프로그램을 사용하기 전에 즉, 248페이지의 "업데이트 프로그램 초기화"에서 설명한 대로 업데이트 프로그램 객체의 initialize() 메서드를 호출하기 전에 이 속성을 정의해야 합니다.

- delay - 숫자입니다. 업데이트를 확인하는 시간 간격(일)입니다. 예를 들어 0.25와 같은 값을 사용할 수 있습니다. 값이 0(기본값)이면 업데이트 프로그램이 자동 정기 확인을 수행하지 않습니다.

ApplicationUpdaterUI의 구성 파일에는 updateURL 및 delay 요소와 함께 다음 요소가 포함될 수 있습니다.

- defaultUI: dialog 요소 목록입니다. 각 dialog 요소에는 사용자 인터페이스의 대화 상자에 해당하는 name 특성이 있습니다. 각 dialog 요소에는 대화 상자가 표시되는지 여부를 정의하는 visible 특성이 있습니다. 기본값은 true입니다. name 특성에 사용할 수 있는 값은 다음과 같습니다.
  - "checkForUpdate" - 업데이트 확인, 업데이트 없음 및 업데이트 오류 대화 상자에 해당합니다.
  - "downloadUpdate" - 업데이트 다운로드 대화 상자에 해당합니다.
  - "downloadProgress" - 다운로드 진행률 및 다운로드 오류 대화 상자에 해당합니다.
  - "installUpdate" - 업데이트 설치 대화 상자에 해당합니다.
  - "fileUpdate" - 파일 업데이트, 파일 업데이트 없음 및 파일 오류 대화 상자에 해당합니다.
- "unexpectedError" - 예기치 않은 오류 대화 상자에 해당합니다.

  false로 설정할 경우 해당하는 대화 상자가 업데이트 절차의 일부로 표시되지 않습니다.

다음은 ApplicationUpdater 프레임워크의 구성 파일 예제입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://ns.adobe.com/air/framework/update/configuration/1.0">
      <url>http://example.com/updates/update.xml</url>
      <delay>1</delay>
</configuration>
```

다음은 defaultUI 요소에 대한 정의를 포함하는 ApplicationUpdaterUI 프레임워크의 구성 파일 예제입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://ns.adobe.com/air/framework/update/configuration/1.0">
      <url>http://example.com/updates/update.xml</url>
      <delay>1</delay>
      <defaultUI>
         <dialog name="checkForUpdate" visible="false" />
         <dialog name="downloadUpdate" visible="false" />
         <dialog name="downloadProgress" visible="false" />
      </defaultUI>
</configuration>
```

configurationFile 속성을 해당 파일의 위치로 지정합니다.

ActionScript 예제:

```
appUpdater.configurationFile = new File("app:/cfg/updateConfig.xml");
```

JavaScript 예제:

```
appUpdater.configurationFile = new air.File("app:/cfg/updateConfig.xml");
```

업데이트 프레임워크의 templates 디렉토리에는 샘플 구성 파일 config-template.xml이 포함됩니다.

업데이트 설정 ActionScript 또는 JavaScript 코드 정의

이러한 구성 매개 변수를 다음과 같이 응용 프로그램의 코드를 사용하여 설정할 수도 있습니다.

```
appUpdater.updateURL = " http://example.com/updates/update.xml";
appUpdater.delay = 1;
```

업데이트 프로그램 객체의 속성은 updateURL 및 delay입니다. 이러한 속성은 구성 파일의 updateURL 및 delay 요소와 동일한 설정 즉, 업데이트 설명자 파일의 URL과 업데이트 확인 간격을 정의합니다. 구성 파일 및 코드의 설정을 지정하는 경우 코드를 사용하여 설정한 속성이 구성 파일의 해당하는 설정보다 우선합니다.

구성 파일이나 스크립트를 통해 updateURL 속성을 정의(244페이지의 “업데이트 설명자 파일 정의 및 웹 서버에 AIR 파일 추가” 참조)한 후에 업데이트 프로그램을 사용(업데이트 프로그램 객체의 initialize() 메서드를 호출(248페이지의 “업데이트 프로그램 초기화”의 설명 참조))해야 합니다.

ApplicationUpdaterUI 프레임워크는 업데이트 프로그램 객체에 대한 다음과 같은 추가 속성을 정의합니다.

- isCheckForUpdateVisible - 업데이트 확인, 업데이트 없음 및 업데이트 오류 대화 상자에 해당합니다.
- isDownloadUpdateVisible - 업데이트 다운로드 대화 상자에 해당합니다.
- isDownloadProgressVisible - 다운로드 진행률 및 다운로드 오류 대화 상자에 해당합니다.
- isInstallUpdateVisible - 업데이트 설치 대화 상자에 해당합니다.
- isFileUpdateVisible - 파일 업데이트, 파일 업데이트 없음 및 파일 오류 대화 상자에 해당합니다.
- isUnexpectedErrorVisible - 예기치 않은 오류 대화 상자에 해당합니다.

각 속성은 ApplicationUpdaterUI 사용자 인터페이스에 있는 하나 이상의 대화 상자에 해당합니다. 각 속성은 기본값이 true인 부울 값입니다. false로 설정할 경우 해당하는 대화 상자가 업데이트 절차의 일부로 표시되지 않습니다.

이러한 대화 상자 속성은 업데이트 구성 파일의 설정을 재정의합니다.

업데이트 프로세스

AIR 업데이트 프레임워크에서는 다음 단계에 따라 업데이트 프로세스를 완료합니다.

1. 업데이트 프로그램 초기화에서 업데이트 확인이 정의된 지연 시간 간격 내에 수행되었는지 확인합니다(245페이지의 “업데이트 설정 구성” 참조). 업데이트 확인이 예정되어 있는 경우 업데이트 프로세스가 계속됩니다.
2. 업데이트 프로그램에서 업데이트 설명자 파일을 다운로드하고 해석합니다.
3. 업데이트 프로그램이 업데이트 AIR 파일을 다운로드합니다.
4. 업데이트 프로그램이 업데이트된 응용 프로그램 버전을 설치합니다.

업데이트 프로그램 객체는 이러한 각 단계 완료 시 이벤트를 전달합니다. ApplicationUpdater 버전에서 프로세스 단계의 성공적인 완료를 알려 주는 이벤트를 취소할 수 있습니다. 이러한 이벤트 중 하나를 취소하려면 프로세스의 다음 단계는 취소됩니다. ApplicationUpdaterUI 버전에서 업데이트 프로그램은 사용자가 프로세스의 각 단계에서 취소 또는 진행 여부를 선택할 수 있는 대화 상자를 표시합니다.

이벤트를 취소하는 경우 업데이트 프로그램 객체의 메서드를 호출하여 프로세스를 다시 시작할 수 있습니다.

업데이트 프로그램의 ApplicationUpdater 버전에서 업데이트 프로세스를 진행하는 동안 현재 상태를 currentState 속성에 기록합니다. 이 속성은 다음과 같은 값의 문자열로 설정됩니다.

- "UNINITIALIZED" - 업데이트 프로그램이 초기화되지 않았습니다.
- "INITIALIZING" - 업데이트 프로그램이 초기화되고 있습니다.
- "READY" - 업데이트 프로그램이 초기화되었습니다
- "BEFORE_CHECKING" - 업데이트 프로그램이 업데이트 설명자 파일을 아직 확인하지 않았습니다.
- "CHECKING" - 업데이트 프로그램이 업데이트 설명자 파일을 확인하고 있습니다.
- "AVAILABLE" - 업데이트 설명자 파일을 사용할 수 있습니다.
- "DOWNLOADING" - 업데이트 프로그램이 AIR 파일을 다운로드하고 있습니다.
- "DOWNLOADED" - 업데이트 프로그램이 AIR 파일을 다운로드했습니다.

- "INSTALLING" - 업데이트 프로그램이 AIR 파일을 설치하고 있습니다.
- "PENDING_INSTALLING" - 업데이트 프로그램이 초기화되었고 대기 중인 업데이트가 있습니다.

업데이트 프로그램 객체의 일부 메서드는 업데이트 프로그램이 특정 상태에 있는 경우 경우에만 실행됩니다.

업데이트 프로그램 초기화

구성 속성을 설정한 후(242페이지의 "기본 예제: ApplicationUpdaterUI 버전 사용" 참조) initialize() 메서드를 호출하여 업데이트를 초기화합니다.

```
appUpdater.initialize();
```

이 메서드는 다음과 같은 작업을 수행합니다.

- 대기 중인 업데이트를 동기적으로 자동 설치하여 업데이트 프레임워크를 초기화합니다. 이 메서드를 호출할 때 응용 프로그램을 다시 시작할 수 있으므로 응용 프로그램 시작 시 이 메서드를 호출해야 합니다.
- 연기된 업데이트가 있는지 확인한 다음 설치합니다.
- 업데이트 프로세스 동안 오류가 발생하는 경우 응용 프로그램 저장소 영역에서 업데이트 파일 및 버전 정보를 지웁니다.
- 지연 시간이 만료되는 경우 업데이트 프로세스를 시작합니다. 그렇지 않은 경우 타이머를 다시 시작합니다.

이 메서드를 호출하면 업데이트 프로그램 객체에서 다음 이벤트를 전달합니다.

- UpdateEvent.INITIALIZED - 초기화가 완료될 때 전달됩니다.
- ErrorEvent.ERROR - 초기화 동안 오류가 발생하는 경우 전달됩니다.

UpdateEvent.INITIALIZED 이벤트가 전달되면 업데이트 프로세스가 완료됩니다.

initialize() 메서드를 호출하면 업데이트 프로그램이 업데이트 프로세스를 시작하고 타이머 지연 시간 설정에 따라 모든 단계를 완료합니다. 그러나 언제든 업데이트 프로그램 객체의 checkNow() 메서드를 호출하여 업데이트 프로세스를 시작할 수 있습니다.

```
appUpdater.checkNow();
```

업데이트 프로세스가 이미 실행되고 있는 경우 이 메서드는 아무 작업도 수행하지 않습니다. 그렇지 않으면 업데이트 프로세스를 시작합니다.

업데이트 프로그램 객체는 checkNow() 메서드 호출의 결과로 다음 이벤트를 전달합니다.

- UpdateEvent.CHECK_FOR_UPDATE 이벤트 - 업데이트 설명자 파일을 다운로드하기 바로 전에 전달됩니다.

checkForUpdate 이벤트를 취소하는 경우 업데이트 프로그램 객체의 checkForUpdate() 메서드를 호출할 수 있습니다. 다음 단원을 참조하십시오. 이벤트를 취소하지 않으면 업데이트 프로세스에서 업데이트 설명자 파일을 확인하는 과정을 계속합니다.

ApplicationUpdaterUI 버전에서 업데이트 프로세스 관리

ApplicationUpdaterUI 버전에서 사용자는 사용자 인터페이스의 대화 상자에서 취소 버튼을 통해 프로세스를 취소할 수 있습니다. 또한 ApplicationUpdaterUI 객체의 cancelUpdate() 메서드를 호출하여 업데이트 프로세스를 프로그래밍 방식으로 취소할 수 있습니다.

ApplicationUpdaterUI 객체의 속성을 설정하거나 업데이트 구성 파일의 요소를 정의하여 업데이트 프로그램에서 표시하는 대화 상자 확인을 지정할 수 있습니다. 자세한 내용은 245페이지의 "업데이트 설정 구성"을 참조하십시오.

ApplicationUpdater 버전에서 업데이트 프로세스 관리

ApplicationUpdater 객체에서 전달한 이벤트의 preventDefault() 메서드를 호출하여 업데이트 프로세스의 단계를 취소할 수 있습니다(247페이지의 "업데이트 프로세스" 참조). 기본 비헤이비어를 취소하면 응용 프로그램에서 사용자에게 계속할지 묻는 메시지를 표시할 수 있게 됩니다.

다음 단원에서는 프로세스 단계가 취소된 경우 업데이트 프로세스를 계속하는 방법에 대해 설명합니다.

업데이트 설명자 파일 다운로드 및 해석

업데이트 프로그램에서 업데이트 설명자 파일을 다운로드하기 직전에 업데이트 프로세스가 시작되기 전에 ApplicationUpdater 객체에서 checkForUpdate 이벤트를 전달합니다. checkForUpdate 이벤트의 기본 비헤이비어를 취소하는 경우 업데이트 프로그램은 업데이트 설명자 파일을 다운로드하지 않습니다. 다음과 같이 checkForUpdate() 메서드를 호출하여 업데이트 프로세스를 다시 시작할 수 있습니다.

```
appUpdater.checkForUpdate();
```

checkForUpdate() 메서드를 호출하면 업데이트 프로그램이 업데이트 설명자 파일을 비동기적으로 다운로드하여 해석합니다. checkForUpdate() 메서드 호출의 결과로 업데이트 프로그램 객체에서 다음 이벤트를 전달할 수 있습니다.

- StatusUpdateEvent.UPDATE_STATUS - 업데이트 프로그램이 업데이트 설명자 파일을 성공적으로 다운로드하고 해석했습니다. 이 이벤트에는 다음과 같은 속성이 있습니다.
  - available - 부울 값입니다. 현재 응용 프로그램의 버전과 다른 버전을 사용할 수 있는 경우 true이고, 그렇지 않은 경우(버전이 동일한 경우) false입니다.
  - version - 문자열입니다. 업데이트 파일의 응용 프로그램 설명자 파일의 버전입니다.
  - details - 배열입니다. 설명에 대한 지역화된 버전이 없는 경우 이 배열은 빈 문자열("")을 첫 번째 요소로, 설명을 두 번째 요소로 반환합니다.

    업데이트 설명자 파일에 여러 버전의 설명이 있는 경우 이 배열에는 여러 하위 배열이 포함됩니다. 각 배열에는 두 요소가 포함되며, 첫째는 언어 코드(예: "en")이고 둘째는 언어에 해당하는 설명(문자열)입니다. 244페이지의 “업데이트 설명자 파일 정의 및 웹 서버에 AIR 파일 추가”를 참조하십시오.
- StatusUpdateErrorEvent.UPDATE_ERROR - 오류가 발생했고 업데이트 프로그램에서 업데이트 설명자 파일을 다운로드 또는 해석하지 못했습니다.

업데이트 AIR 파일 다운로드

업데이트 프로그램에서 업데이트 설명자 파일을 성공적으로 다운로드 및 해석한 후 ApplicationUpdater 객체는 updateStatus 이벤트를 전달합니다. 기본 비헤이비어는 사용 가능한 업데이트가 있는 경우 다운로드를 시작하는 것입니다. 기본 비헤이비어를 취소하는 경우 downloadUpdate() 메서드를 호출하여 업데이트 프로세스를 다시 시작할 수 있습니다.

```
appUpdater.downloadUpdate();
```

이 메서드를 호출하면 업데이트 프로그램에서 AIR 파일의 업데이트 버전을 비동기적으로 다운로드합니다.

downloadUpdate() 메서드는 다음 이벤트를 전달할 수 있습니다.

- UpdateEvent.DOWNLOAD_START - 서버에 대한 연결이 설정되었습니다. ApplicationUpdaterUI 라이브러리를 사용할 경우 이 이벤트는 다운로드 진행률을 추적하는 진행률 막대가 있는 대화 상자를 표시합니다.
- ProgressEvent.PROGRESS - 파일 다운로드가 진행될 때 주기적으로 전달됩니다.
- DownloadErrorEvent.DOWNLOAD_ERROR - 업데이트 파일을 연결하거나 다운로드하는 동안 오류가 발생하는 경우 전달됩니다. HTTP 상태가 잘못된 경우에도 전달됩니다(예: "404 - 파일을 찾을 수 없습니다”). 이 이벤트에는 추가 오류 정보를 정의하는 정수인 errorID 속성이 있습니다. 추가 subErrorID 속성에는 추가 오류 정보가 포함될 수 있습니다.
- UpdateEvent.DOWNLOAD_COMPLETE - 업데이트 프로그램이 업데이트 설명자 파일을 성공적으로 다운로드하고 해석했습니다. 이 이벤트를 취소하지 않으면 ApplicationUpdater 버전이 업데이트 버전 설치를 진행합니다. ApplicationUpdaterUI 버전에서는 계속하기 위한 옵션을 제공하는 대화 상자를 표시합니다.

응용 프로그램 업데이트

업데이트 다운로드가 완료되면 ApplicationUpdater 객체는 downloadComplete 이벤트를 전달합니다. 기본 비헤이비어를 취소하는 경우 installUpdate() 메서드를 호출하여 업데이트 프로세스를 다시 시작할 수 있습니다.

```
appUpdater.installUpdate(file);
```

이 메서드를 호출하면 업데이트 프로그램은 AIR 파일의 업데이트 버전을 설치합니다. 이 메서드에는 업데이트 사용할 AIR 파일을 참조하는 File 객체인 file 매개 변수 하나가 포함되어 있습니다.

installUpdate() 메서드 호출의 결과로 ApplicationUpdater 객체가 beforeInstall 이벤트를 전달할 수 있습니다.

- UpdateEvent.BEFORE_INSTALL - 업데이트를 설치하기 직전에 전달됩니다. 경우에 따라 사용자가 업데이트를 계속하기 전에 현재 작업을 완료할 수 있도록 지금 업데이트를 설치하지 않도록 하는 데 이 이벤트를 사용할 수 있습니다. Event 객체의 preventDefault() 메서드를 호출하면 다음 재시작 시까지 설치가 연기되고 추가 업데이트 프로세스를 시작할 수 없습니다. 여기에는 checkNow() 메서드 호출의 결과나 정기적인 확인으로 인해 발생하는 업데이트도 포함됩니다.

임의의 AIR 파일에서 설치

installFromAIRFile() 메서드를 호출하여 사용자 컴퓨터에 있는 AIR 파일의 업데이트 버전을 설치할 수 있습니다.

```
appUpdater.installFromAIRFile();
```

이 메서드를 호출하면 업데이트 프로그램은 AIR 파일의 업데이트 버전을 설치합니다.

installFromAIRFile() 메서드는 다음 이벤트를 전달할 수 있습니다.

- StatusFileUpdateEvent.FILE_UPDATE_STATUS - ApplicationUpdater에서 installFromAIRFile() 메서드를 사용하여 전송한 파일의 유효성을 검사한 후 전달됩니다. 이 이벤트에는 다음과 같은 속성이 있습니다.
  - available - 현재 응용 프로그램 버전과 다른 버전을 사용할 수 있는 경우 true이고 그렇지 않은 경우(버전이 동일한 경우) false입니다.
  - version - 사용할 수 있는 새 버전을 나타내는 문자열입니다.
  - path - 업데이트 파일의 기본 경로를 나타냅니다.

  StatusFileUpdateEvent 객체의 available 속성을 true로 설정한 경우 이 이벤트를 취소할 수 있습니다. 이 이벤트를 취소하면 업데이트 진행이 취소됩니다. 취소된 업데이트를 계속하려면 installUpdate() 메서드를 호출합니다.
- StatusFileUpdateErrorEvent.FILE_UPDATE_ERROR - 오류가 발생하여 업데이트 프로그램이 AIR 응용 프로그램을 설치하지 못했습니다.

업데이트 프로세스 취소

cancelUpdate() 메서드를 호출하여 업데이트 프로세스를 취소할 수 있습니다.

```
appUpdater.cancelUpdate();
```

이 메서드는 완료되지 않은 다운로드 파일을 삭제하여 대기 중인 다운로드를 취소하고 정기 확인 타이머를 다시 시작합니다.

업데이트 프로그램 객체가 초기화되고 있는 경우 이 메서드는 아무 작업도 수행하지 않습니다.

ApplicationUpdaterUI 인터페이스 지역화

ApplicationUpdaterUI 클래스에서는 업데이트 프로세스를 위한 기본 사용자 인터페이스를 제공합니다. 여기에는 사용자가 프로세스를 시작하고, 프로세스를 취소하고, 기타 관련 작업을 수행할 수 있는 대화 상자가 포함됩니다.

업데이트 설명자 파일의 description 요소를 사용하면 응용 프로그램에 대한 설명을 여러 언어로 정의할 수 있습니다. 다음과 같이 lang 특성을 정의하는 여러 text 요소를 사용합니다.

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1a1</version>
       <url>http://example.com/updates/sample_1.1a1.air</url>
       <description>
           <text xml:lang="en">English description</text>
           <text xml:lang="fr">French description</text>
           <text xml:lang="ro">Romanian description</text>
       </description>
    </update>
```

업데이트 프레임워크에서는 최종 사용자의 지역화 체인에 가장 잘 맞는 설명을 사용합니다. 자세한 내용은 업데이트 설명자 파일 정의 및 웹 서버에 AIR 파일 추가를 참조하십시오.

Flex 개발자는 "ApplicationUpdaterDialogs" 번들에 새 언어를 직접 추가할 수 있습니다.

JavaScript 개발자는 업데이트 프로그램 객체의 addResources() 메서드를 호출할 수 있습니다. 이 메서드는 언어에 대한 새 리소스 번들을 동적으로 추가합니다. 리소스 번들은 언어에 대한 지역화된 문자열을 정의합니다. 이러한 문자열은 다양한 대화 상자 텍스트 필드에서 사용됩니다.

JavaScript 개발자는 ApplicationUpdaterUI 클래스의 localeChain 속성을 사용하여 사용자 인터페이스에서 사용되는 로캘 체인을 정의할 수 있습니다. 일반적으로 JavaScript(HTML) 개발자만 이 속성을 사용합니다. Flex 개발자는 ResourceManager를 사용하여 로캘 체인을 관리합니다.

예를 들어 다음 JavaScript 코드는 루마니아어와 헝가리어의 리소스 번들을 정의합니다.

```
appUpdater.addResources("ro_RO",
                {titleCheck: "Titlu", msgCheck: "Mesaj", btnCheck: "Buton"});
appUpdater.addResources("hu", {titleCheck: "Cím", msgCheck: "Üzenet"});
var languages = ["ro", "hu"];
languages = languages.concat(air.Capabilities.languages);
var sortedLanguages = air.Localizer.sortLanguagesByPreference(languages,
                     air.Capabilities.language,
                     "en-US");
sortedLanguages.push("en-US");
appUpdater.localeChain = sortedLanguages;
```

자세한 내용은 언어 참조 설명서에서 ApplicationUpdaterUI 클래스의 addResources() 메서드에 대한 설명을 참조하십시오.

18장: 소스 코드 보기

웹 브라우저에서 HTML 페이지의 소스 코드를 볼 수 있는 것과 마찬가지로 HTML 기반 AIR 응용 프로그램의 소스 코드를 볼 수 있습니다. Adobe® AIR® SDK에는 응용 프로그램에서 최종 사용자에게 소스 코드를 보여 줄 때 사용할 수 있는 AIRSourceViewer.js라는 JavaScript 파일이 포함되어 있습니다.

소스 뷰어 로드, 구성 및 열기

소스 뷰어 코드는 JavaScript 파일인 AIRSourceViewer.js에 포함되어 있으며, 이 파일은 AIR SDK의 frameworks 디렉토리에 있습니다. 응용 프로그램에서 소스 뷰어를 사용하려면 AIRSourceViewer.js를 응용 프로그램 프로젝트 디렉토리에 복사하고 응용 프로그램의 기본 HTML 파일에서 script 태그를 사용하여 로드합니다.

```
<script type="text/javascript" src="AIRSourceViewer.js"></script>
```

AIRSourceViewer.js 파일은 SourceViewer 클래스를 정의합니다. JavaScript 코드에서는 air.SourceViewer를 호출하여 이 클래스에 액세스할 수 있습니다.

SourceViewer 클래스는 getDefault(), setup() 및 viewSource()라는 세 가지 메서드를 정의합니다.

메서드	설명
getDefault()	정적 메서드입니다. 다른 메서드를 호출하는 데 사용할 수 있는 SourceViewer 인스턴스를 반환합니다.
setup()	소스 뷰어에 구성 설정을 적용합니다. 자세한 내용은 252페이지의 "소스 뷰어 구성"을 참조하십시오.
viewSource()	사용자가 호스트 응용 프로그램의 소스 파일을 찾아보고 열 수 있는 새 윈도우를 엽니다.

참고: 소스 뷰어를 사용하는 코드는 응용 프로그램 보안 샌드박스(응용 프로그램 디렉토리의 파일)에 있어야 합니다.

예를 들어, 다음 JavaScript 코드는 소스 뷰어 객체를 인스턴스화하고 모든 소스 파일을 나열하는 소스 뷰어 윈도우를 엽니다.

```
var viewer = air.SourceViewer.getDefault();
viewer.viewSource();
```

소스 뷰어 구성

config() 메서드는 지정된 설정을 소스 뷰어에 적용합니다. 이 메서드는 configObject 매개 변수 하나만 사용합니다. configObject 객체에는 소스 뷰어에 대한 구성 설정을 정의하는 속성이 있습니다. 속성은 default, exclude, initialPosition, modal, typesToRemove 및 typesToAdd입니다.

default

소스 뷰어에 표시할 초기 파일에 대한 상대 경로를 지정하는 문자열입니다.

예를 들어, 다음 JavaScript 코드는 index.html 파일이 초기 파일로 표시된 상태로 소스 뷰어 윈도우를 엽니다.

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.default = "index.html";
viewer.viewSource(configObj);
```

exclude

소스 뷰어 목록에서 제외할 파일이나 디렉토리를 지정하는 문자열 배열입니다. 경로는 응용 프로그램 디렉토리에 대해 상대적입니다. 와일드카드 문자는 사용할 수 없습니다.

예를 들어, 다음 JavaScript 코드는 소스 뷰어 윈도우를 열고 AIRSourceViewer.js 파일과 Images 및 Sounds 하위 디렉토리의 파일을 제외한 모든 소스 파일을 나열합니다.

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.exclude = ["AIRSourceViewer.js", "Images" "Sounds"];
viewer.viewSource(configObj);
```

initialPosition

소스 뷰어 윈도우의 초기 x 및 y 좌표를 지정하는 숫자 두 개를 포함하는 배열입니다.

예를 들어, 다음 JavaScript 코드는 화면 좌표 [40, 60](X = 40, Y = 60)에 소스 뷰어 윈도우를 엽니다.

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.initialPosition = [40, 60];
viewer.viewSource(configObj);
```

modal

소스 뷰어 윈도우가 모달 윈도우(true)인지 비 모달 윈도우(false)인지를 지정하는 부울 값입니다. 기본적으로 소스 뷰어 윈도우는 모달 윈도우입니다.

예를 들어, 다음 JavaScript 코드는 사용자가 소스 뷰어 윈도우와 다른 응용 프로그램 윈도우를 동시에 사용할 수 있도록 소스 뷰어 윈도우를 엽니다.

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.modal = false;
viewer.viewSource(configObj);
```

typesToAdd

기본 유형과 함께 소스 뷰어 목록에 포함될 파일 유형을 지정하는 문자열 배열입니다.

기본적으로 소스 뷰어 윈도우는 다음 파일 유형을 나열합니다.

- 텍스트 파일 - TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- 이미지 파일 - JPG, JPEG, PNG, GIF

  값을 지정하지 않으면 typesToExclude 속성에 지정된 유형을 제외한 모든 기본 유형이 포함됩니다.

  예를 들어, 다음 JavaScript 코드는 VCF와 VCARD 파일이 포함된 소스 뷰어 윈도우를 엽니다.

  ```
  var viewer = air.SourceViewer.getDefault();
  var configObj = {};
  configObj.typesToAdd = ["text.vcf", "text.vcard"];
  viewer.viewSource(configObj);
  ```

  나열하는 각 파일 유형에 대해 "text"(텍스트 파일 유형) 또는 "image"(이미지 파일 유형)를 지정해야 합니다.

typesToExclude

소스 뷰어에서 제외할 파일 유형을 지정하는 문자열 배열입니다.

기본적으로 소스 뷰어 윈도우는 다음 파일 유형을 나열합니다.

- 텍스트 파일 - TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- 이미지 파일 - JPG, JPEG, PNG, GIF

  예를 들어, 다음 JavaScript 코드는 GIF나 XML 파일을 나열하지 않도록 소스 뷰어 윈도우를 엽니다.

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToExclude = ["image.gif", "text.xml"];
viewer.viewSource(configObj);
```

나열하는 각 파일 유형에 대해 "text"(텍스트 파일 유형) 또는 "image"(이미지 파일 유형)를 지정해야 합니다.

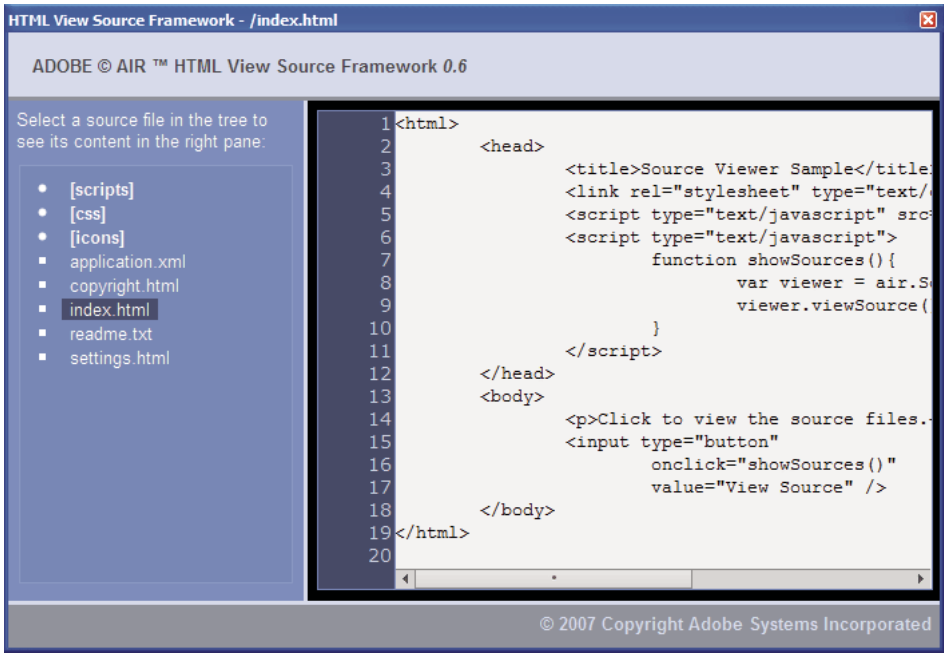
소스 뷰어 열기

사용자가 선택하여 소스 뷰어 코드를 호출할 수 있는 링크, 버튼, 메뉴 명령 등과 같은 사용자 인터페이스 요소가 있어야 합니다. 예를 들어, 다음은 사용자가 링크를 클릭하면 소스 뷰어를 여는 간단한 응용 프로그램입니다.

```
<html>
    <head>
        <title>Source Viewer Sample</title>
        <script type="text/javascript" src="AIRSourceViewer.js"></script>
        <script type="text/javascript">
            function showSources(){
                var viewer = air.SourceViewer.getDefault();
                viewer.viewSource()
            }
        </script>
    </head>
    <body>
        <p>Click to view the source files.</p>
        <input type="button"
            onclick="showSources()"
            value="View Source" />
    </body>
</html>
```

소스 뷰어 사용자 인터페이스

응용 프로그램에서 SourceViewer 객체의 viewSource() 메서드를 호출하면 AIR 응용 프로그램이 소스 뷰어 윈도우를 엽니다. 이 윈도우 왼쪽에는 소스 파일 및 디렉토리 목록이 있고 오른쪽에는 선택한 파일의 소스 코드를 표시하는 디스플레이 영역이 있습니다.



디렉토리는 대괄호 안에 나열됩니다. 사용자는 대괄호를 클릭하여 디렉토리 목록을 확장하거나 축소할 수 있습니다.

소스 뷰어는 인식된 확장명을 가진 텍스트 파일(, HTML, JS, TXT, XML 등)이나 인식된 이미지 확장명을 가진 이미지 파일(JPG, JPEG, PNG 및 GIF)의 소스를 표시할 수 있습니다. 사용자가 인식된 파일 확장명이 없는 파일을 선택하면 "이 파일 유형에서 텍스트 내용을 검색할 수 없습니다."라는 오류 메시지가 표시됩니다.

setup() 메서드를 통해 제외된 모든 소스 파일은 나열되지 않습니다(252페이지의 “소스 뷰어 로드, 구성 및 열기” 참조).

19장: AIR HTML Introspector를 사용한 디버깅

Adobe® AIR® SDK에는 응용 프로그램에 포함시켜 HTML 기반 응용 프로그램을 디버깅하는 데 사용할 수 있는 AIRIntrospector.js JavaScript 파일이 포함되어 있습니다.

AIR Introspector

Adobe AIR HTML/JavaScript Application Introspector(AIR HTML Introspector)는 HTML 기반 응용 프로그램을 개발하고 디버깅하는 데 도움을 주는 다음과 같은 유용한 기능을 제공합니다.

- 응용 프로그램의 사용자 인터페이스 요소를 가리켜 해당 마크업과 DOM 속성을 확인할 수 있는 검사 도구가 포함되어 있습니다.
- 검사를 위해 객체 참조를 보내거나, 속성 값을 조정하고 JavaScript 코드를 실행할 수 있는 콘솔이 포함되어 있습니다. 또한 객체를 콘솔과 직렬화하여 데이터를 편집하지 못하도록 제한할 수 있습니다. 콘솔에서 텍스트를 복사하고 저장할 수 있습니다.
- DOM 속성 및 함수의 트리 보기가 포함되어 있습니다.
- DOM 요소의 특성과 텍스트 노드를 편집할 수 있습니다.
- 응용 프로그램에 로드된 링크, CSS 스타일, 이미지 및 JavaScript 파일을 목록으로 표시합니다.
- 사용자 인터페이스의 초기 HTML 소스와 현재 마크업 소스를 확인할 수 있습니다.
- 응용 프로그램 디렉토리에 있는 파일에 액세스할 수 있습니다. 이 기능은 응용 프로그램 샌드박스에 대해 열린 AIR HTML Introspector 콘솔에서만 사용할 수 있습니다. 응용 프로그램 샌드박스 내용에서 열린 콘솔이 아닌 경우 이 기능을 사용할 수 없습니다.
- XMLHttpRequest 객체와 responseText, responseXML 등을 비롯한 해당 속성(사용할 수 있는 경우)을 볼 수 있는 뷰어가 포함되어 있습니다.
- 소스 코드와 파일에서 일치하는 텍스트를 검색할 수 있습니다.

AIR Introspector 코드 로드

AIR Introspector 코드는 JavaScript 파일인 AIRIntrospector.js에 포함되어 있으며, 이 파일은 AIR SDK의 frameworks 디렉토리에 있습니다. 응용 프로그램에서 AIR Introspector를 사용하려면 AIRIntrospector.js를 응용 프로그램 프로젝트 디렉토리에 복사하고 응용 프로그램의 기본 HTML 파일에서 script 태그를 사용하여 로드합니다.

```
<script type="text/javascript" src="AIRIntrospector.js"></script>
```

또한 응용 프로그램의 다양한 기본 윈도우에 해당하는 모든 HTML 파일을 파일에 포함시킵니다.

중요: AIRIntrospector.js 파일은 응용 프로그램을 개발하고 디버깅할 때만 포함시키십시오. 배포하려고 패키지화한 AIR 응용 프로그램에서는 이 파일을 제거해야 합니다.

AIRIntrospector.js 파일은 Console 클래스를 정의합니다. JavaScript 코드에서는 air.Introspector.Console을 호출하여 이 클래스에 액세스할 수 있습니다.

참고: AIR Introspector를 사용하는 코드는 응용 프로그램 보안 샌드박스(응용 프로그램 디렉토리의 파일)에 있어야 합니다.

콘솔 탭에서 객체 검사

Console 클래스에는 log(), warn(), info(), error() 및 dump()의 다섯 가지 메서드가 정의되어 있습니다.

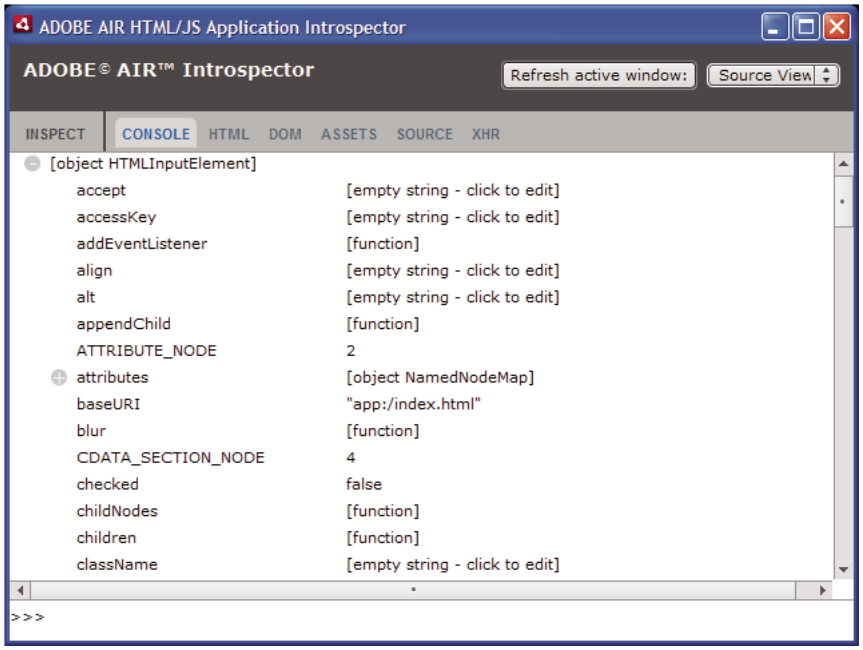
log(), warn(), info() 및 error() 메서드 모두 객체를 콘솔 탭으로 보내는 데 사용할 수 있습니다. 이러한 메서드 중 가장 기본적인 메서드는 log() 메서드입니다. 다음 코드에서는 test 변수로 표현된 단순 객체를 콘솔 탭으로 보냅니다.

```
var test = "hello";
air.Introspector.Console.log(test);
```

하지만 복합 객체를 콘솔 탭으로 보내는 것이 유용한 경우가 많습니다. 예를 들어, 다음 HTML 페이지에는 버튼 객체 자체를 콘솔 탭으로 보내는 함수를 호출한 버튼(btn1)이 포함되어 있습니다.

```
<html>
    <head>
        <title>Source Viewer Sample</title>
        <script type="text/javascript" src="scripts/AIRIntrospector.js"></script>
        <script type="text/javascript">
            function logBtn()
            {
                var button1 = document.getElementById("btn1");
                air.Introspector.Console.log(button1);
            }
        </script>
    </head>
    <body>
        <p>Click to view the button object in the Console.</p>
        <input type="button" id="btn1"
            onclick="logBtn()"
            value="Log" />
    </body>
</html>
```

이 버튼을 클릭하면 콘솔 탭에 btn1 객체가 표시되고, 객체의 트리 보기를 확장하여 속성을 검사할 수 있습니다.



속성 이름 오른쪽에 있는 목록을 클릭하고 텍스트 목록을 수정하여 객체의 속성을 편집할 수 있습니다.

info(), error() 및 warn() 메서드는 log() 메서드와 동일하지만 호출될 때 콘솔 탭의 줄 시작 부분에 다음과 같은 아이콘이 표시된다는 것이 다릅니다.

메서드	아이콘
info()	
error()	
warn()	

log(), warn(), info() 및 error() 메서드는 실제 객체에 대한 참조만 보내므로 사용할 수 있는 속성은 보는 시점의 속성입니다. 실제 객체와 직렬화하고 싶은 경우에는 dump() 메서드를 사용하십시오. 이 메서드에는 매개 변수가 두 개 있습니다.

매개 변수	설명
dumpObject	직렬화할 객체입니다.
levels	객체 트리에서 루트 수준과 함께 검사할 수준의 최대 수입니다. 기본값은 1로, 트리의 루트 수준에서 한 수준 깊은 수준이 표시됩니다. 이 매개 변수는 선택적입니다.

dump() 메서드를 호출하면 객체를 콘솔 탭으로 보내기 전에 직렬화하므로 객체의 속성을 편집할 수 없게 됩니다. 예를 들어 다음과 같은 코드를 살펴봅니다.

```
var testObject = new Object();
testObject.foo = "foo";
testObject.bar = 234;
air.Introspector.Console.dump(testObject);
```

이 코드를 실행하면 콘솔에 testObject 객체와 해당 속성이 나타나지만 콘솔에서 속성 값을 편집할 수 없습니다.

AIR Introspector 구성

AIRIntrospectorConfig 전역 변수를 설정하여 콘솔을 구성할 수 있습니다. 예를 들어, 다음 JavaScript 코드는 100자 단위로 열을 줄 바꿈하도록 AIR Introspector를 구성합니다.

```
var AIRIntrospectorConfig = new Object();
AIRIntrospectorConfig.wrapColumns = 100;
```

AIRIntrospectorConfig 변수의 속성은 script 태그를 통해 AIRIntrospector.js 파일을 로드하기 전에 설정해야 합니다.

AIRIntrospectorConfig 변수에는 다음과 같은 8가지 속성이 있습니다.

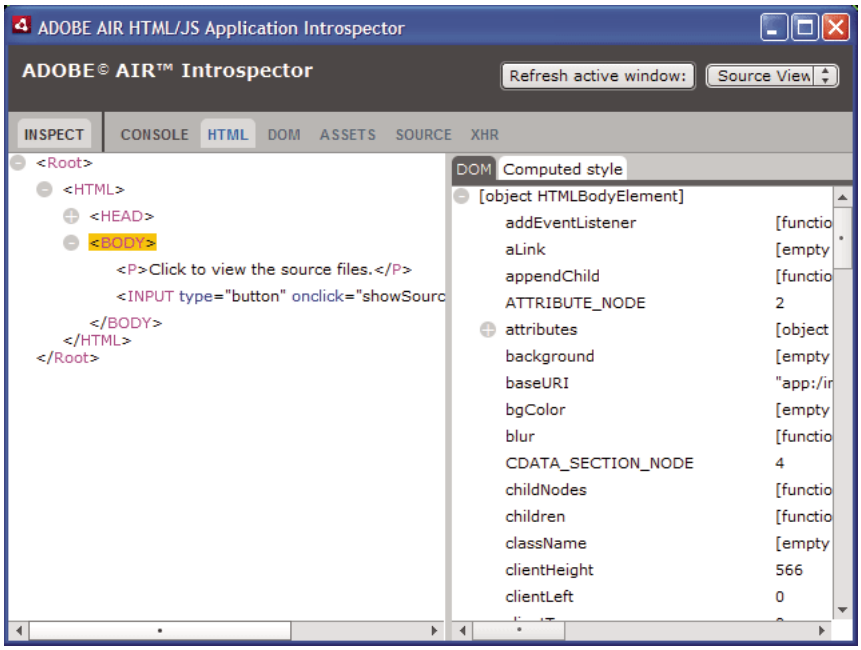
속성	기본값	설명
closeIntrospectorOnExit	true	응용 프로그램의 다른 모든 윈도우가 닫히면 관리자 윈도우가 닫히도록 설정합니다.
debuggerKey	123(F12 키)	AIR Introspector 윈도우를 표시하거나 숨기는 키보드 단축키에 대한 키 코드입니다.
debugRuntimeObjects	true	JavaScript에 정의되어 있는 객체와 함께 런타임 객체를 확장하도록 설정합니다.
flashTabLabels	true	콘솔 및 XMLHttpRequest 탭이 변경되면 깜빡이도록 설정합니다(예: 이러한 탭에서 텍스트가 기록된 경우).
introspectorKey	122(F11 키)	검사 패널을 여는 키보드 단축키에 대한 키 코드입니다.

속성	기본값	설명
showTimestamp	true	콘솔 탭에서 각 줄의 시작 부분에 타임스탬프를 표시하도록 설정합니다.
showSender	true	콘솔 탭에서 각 줄의 시작 부분에 메시지를 보내는 객체에 대한 정보를 표시하도록 설정합니다.
wrapColumns	2000	소스 파일에서 줄 바꿈을 할 열 번호입니다.

AIR Introspector 인터페이스

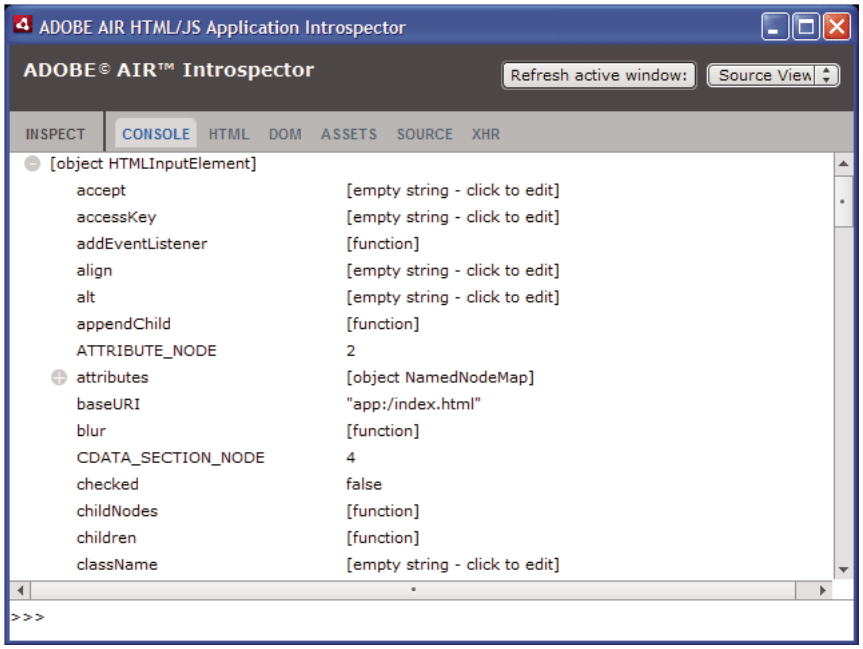
응용 프로그램을 디버깅할 때 AIR introspector 윈도우를 열려면 F12 키를 누르거나 Console 클래스의 메서드 중 하나를 호출합니다(257페이지의 “콘솔 탭에서 객체 검사” 참조). F12 키가 아닌 키를 바로 가기 키로 구성할 수 있습니다. 자세한 내용은 258 페이지의 “AIR Introspector 구성”을 참조하십시오.

다음 그림에서 볼 수 있는 것처럼 AIR Introspector 윈도우에는 콘솔, HTML, DOM, 에셋, 소스 및 XHR의 6개 탭이 있습니다.



콘솔 탭

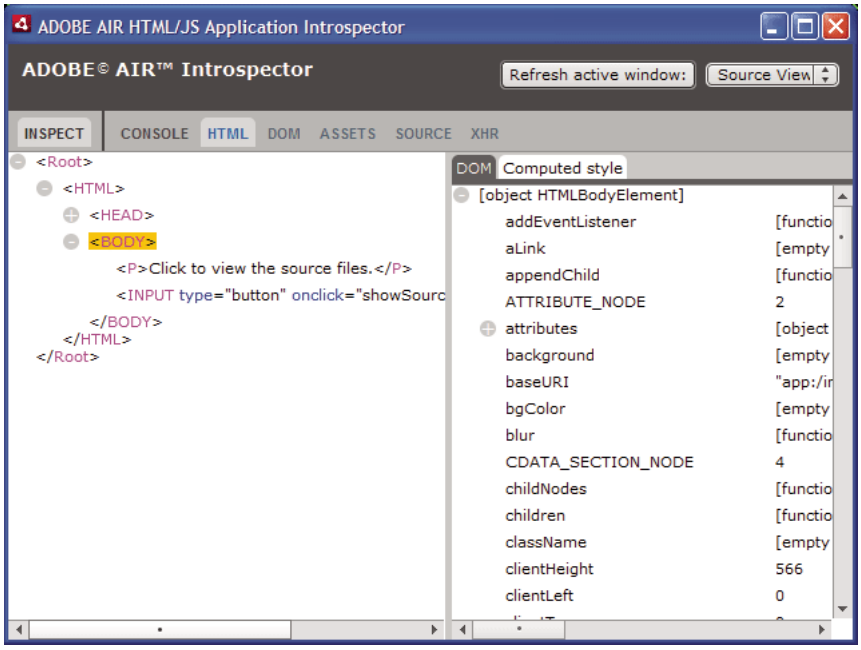
[콘솔] 탭에는 air.Introspector.Console 클래스의 메서드 중 하나에 매개 변수로 전달된 속성 값이 표시됩니다. 자세한 내용은 257페이지의 "콘솔 탭에서 객체 검사"를 참조하십시오.



- 콘솔을 지우려면 텍스트를 마우스 오른쪽 버튼으로 클릭하고 [콘솔 지우기]를 선택합니다.
- [콘솔] 탭에 있는 텍스트를 파일에 저장하려면 콘솔 탭을 마우스 오른쪽 버튼으로 클릭하고 [파일에 콘솔 저장]을 선택합니다.
- [콘솔] 탭에 있는 텍스트를 클립보드에 저장하려면 콘솔 탭을 마우스 오른쪽 버튼으로 클릭하고 [클립보드에 콘솔 저장]을 선택합니다. 선택한 텍스트를 클립보드에 복사하려면 텍스트를 마우스 오른쪽 버튼으로 클릭하고 [복사]를 선택합니다.
- Console 클래스에 있는 텍스트를 파일에 저장하려면 콘솔 탭을 마우스 오른쪽 버튼으로 클릭하고 [파일에 콘솔 저장]을 선택합니다.
- 탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

HTML 탭

[HTML] 탭을 사용하면 전체 HTML DOM을 트리 구조로 볼 수 있습니다. 요소를 클릭하면 탭 오른쪽에 해당 속성이 표시됩니다. 트리의 노드를 확장하거나 축소하려면 + 및 - 아이콘을 클릭합니다.



[HTML] 탭에서는 모든 특성과 텍스트 요소를 편집할 수 있으며, 편집한 값은 응용 프로그램에 반영됩니다.

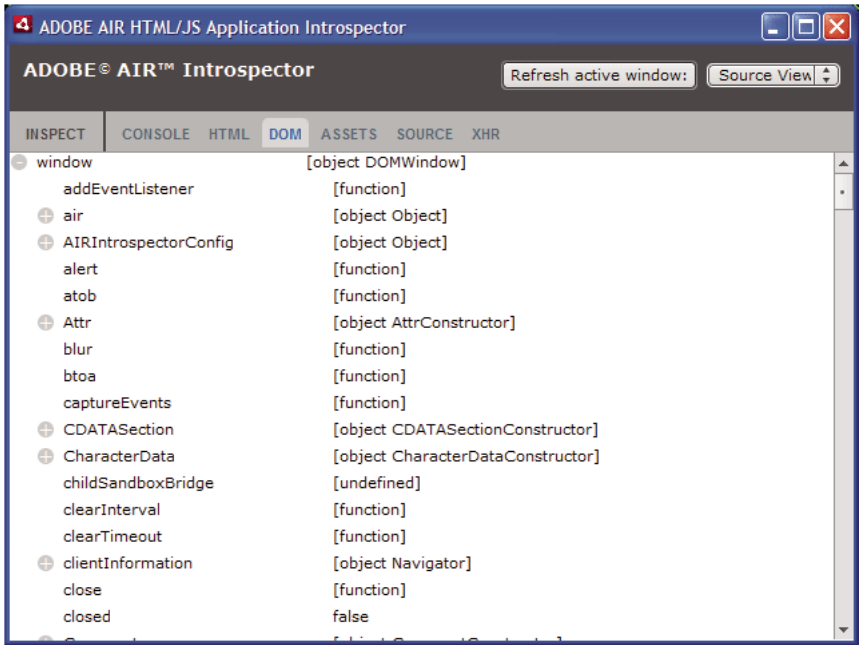
AIR Introspector 윈도우의 탭 목록 왼쪽에 있는 [검사] 버튼을 클릭합니다. 기본 윈도우의 HTML 페이지에서 원하는 요소를 클릭하면 [HTML] 탭에 연결된 DOM 객체가 표시됩니다. 기본 윈도우에 포커스가 있는 경우 키보드 단축키를 눌러 [검사] 버튼을 설정하거나 해제할 수 있습니다. 키보드 단축키는 기본적으로 F11입니다. F11 키가 아닌 키를 키보드 단축키로 구성할 수 있습니다. 자세한 내용은 258페이지의 “AIR Introspector 구성”을 참조하십시오.

[HTML] 탭에 표시된 데이터를 새로 고치려면 AIR Introspector 윈도우 맨 위에 있는 [활성 윈도우 새로 고침] 버튼을 클릭합니다.

탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

DOM 탭

[DOM] 탭에는 윈도우 객체가 트리 구조로 표시됩니다. 모든 문자열 및 숫자 속성을 편집할 수 있으며, 편집한 값은 응용 프로그램에 반영됩니다.

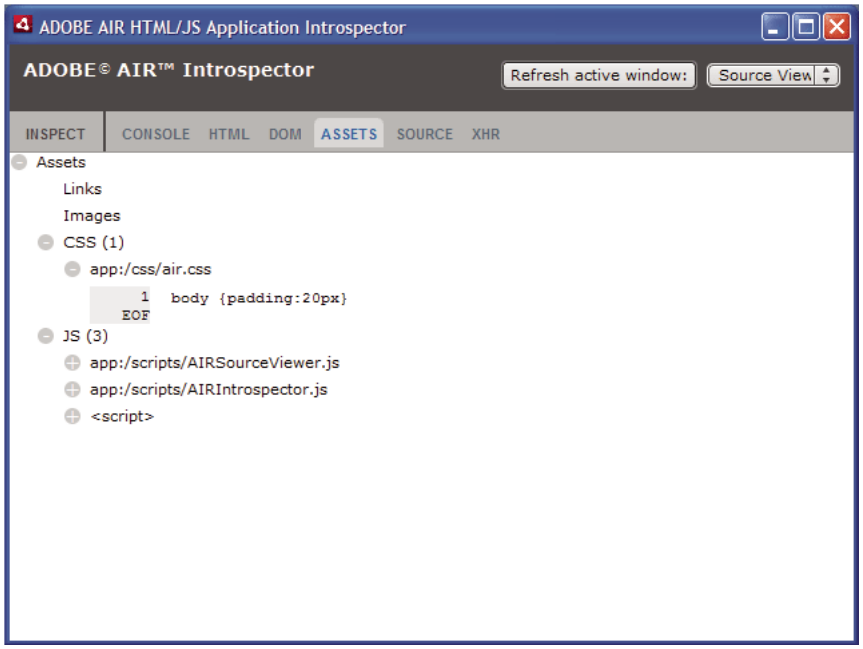


[DOM] 탭에 표시된 데이터를 새로 고치려면 AIR Introspector 윈도우 맨 위에 있는 [활성 윈도우 새로 고침] 버튼을 클릭합니다.

탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

에셋 탭

[에셋] 탭을 사용하면 기본 윈도우에 로드된 링크, 이미지, CSS 및 JavaScript 파일을 확인할 수 있습니다. 이러한 노드 중 하나를 확장하면 파일의 내용이나 실제 사용된 이미지가 표시됩니다.



[에셋] 탭에 표시된 데이터를 새로 고치려면 AIR Introspector 윈도우 맨 위에 있는 [활성 윈도우 새로 고침] 버튼을 클릭합니다.

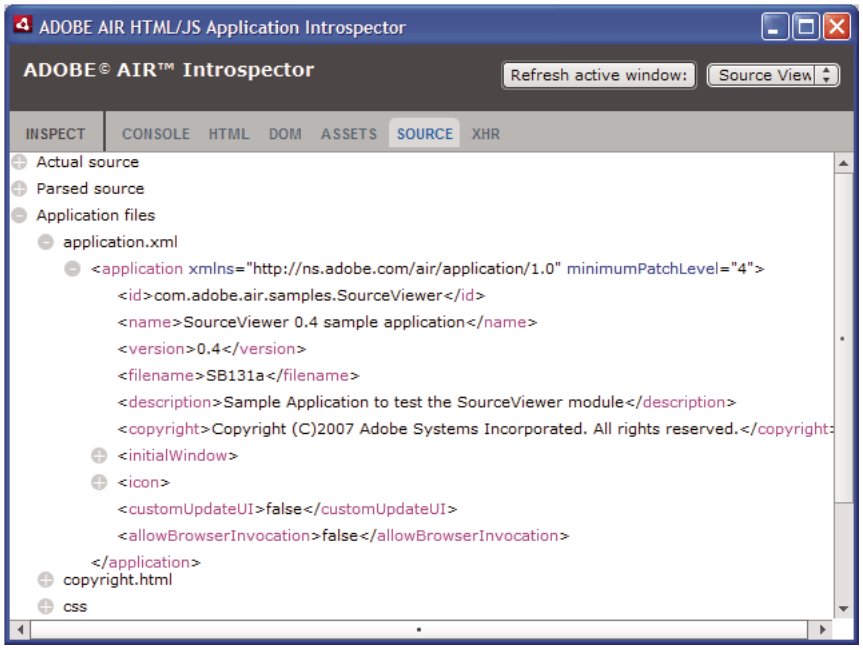
탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

소스 탭

[소스] 탭에는 섹션 세 개가 있습니다.

- 실제 소스 - 응용 프로그램을 시작할 때 루트 내용으로 로드된 페이지의 HTML 소스를 표시합니다.
- 파싱된 소스 - 응용 프로그램 UI를 구성하는 현재 마크업을 표시합니다. 응용 프로그램은 Ajax 기술을 사용하여 즉석에서 마크업 코드를 생성하므로 마크업 코드가 실제 소스와 다를 수 있습니다.

- 응용 프로그램 파일 - 응용 프로그램 디렉토리에 있는 파일을 나열합니다. 이 목록은 응용 프로그램 보안 샌드박스의 내용에서 실행된 AIR Introspector에서만 사용할 수 있습니다. 이 섹션에서 텍스트 파일의 내용이나 이미지를 볼 수 있습니다.

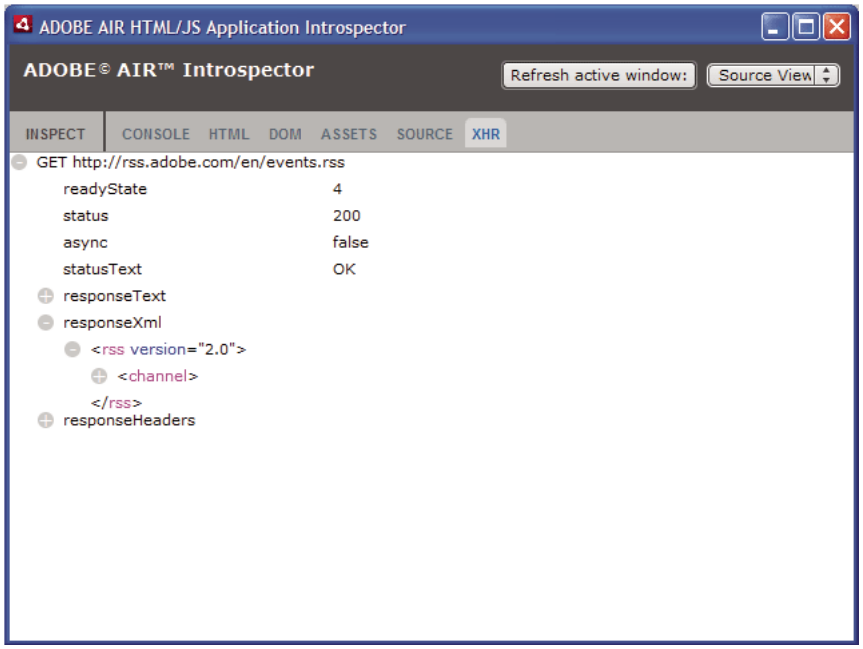


[소스] 탭에 표시된 데이터를 새로 고치려면 AIR Introspector 윈도우 맨 위에 있는 [활성 윈도우 새로 고침] 버튼을 클릭합니다.

탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

XHR 탭

[XHR] 탭은 응용 프로그램의 모든 XMLHttpRequest 통신을 가로채 정보를 기록합니다. 이 탭을 사용하면 responseText, responseXML 등을 비롯한 XMLHttpRequest 속성(사용 가능한 경우)을 트리 보기로 볼 수 있습니다.



탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

비 응용 프로그램 샌드박스의 내용으로 AIR Introspector 사용

응용 프로그램 디렉토리의 내용을 비 응용 프로그램 샌드박스에 매핑된 iframe이나 frame으로 로드할 수 있습니다. Adobe AIR의 HTML 보안(ActionScript 개발자용) 또는 Adobe AIR의 HTML 보안(HTML 개발자용)을 참조하십시오. 이러한 내용을 포함한 AIR introspector를 사용할 수 있지만 다음과 같은 규칙을 따라야 합니다.

- AIRIntrospector.js 파일이 응용 프로그램 샌드박스 내용과 비 응용 프로그램 샌드박스(iframe) 내용에 모두 포함되어야 합니다.
- parentSandboxBridge 속성은 덮어쓰지 마십시오. AIR Introspector 코드에서 이 속성을 사용합니다. 필요한 경우에는 속성을 추가합니다. 예를 들어, 다음과 같이 하는 대신

  ```
  parentSandboxBridge = mytrace: function(str) {runtime.trace(str)} ;
  ```

  다음과 같은 구문을 사용하십시오.

  ```
  parentSandboxBridge.mytrace = function(str) {runtime.trace(str)};
  ```

- 비 응용 프로그램 샌드박스 내용에서는 F12 키를 누르거나 air.Introspector.Console 클래스의 메서드 중 하나를 호출하여 AIR Introspector를 열 수 없습니다. 이 경우 Introspector 윈도우를 열려면 [Introspector 열기] 버튼을 클릭해야 합니다. 이 버튼은 기본적으로 iframe 또는 frame의 맨 오른쪽 위에 추가됩니다. 비 응용 프로그램 샌드박스 내용에 적용되는 보안 제한 때문에 버튼 클릭과 같은 사용자 동작의 결과로만 새 윈도우를 열 수 있습니다.

- 응용 프로그램 샌드박스와 비 응용 프로그램 샌드박스 각각에 대해 별도의 AIR Introspector 윈도우를 열 수 있습니다. 두 윈도우는 AIR Introspector 윈도우에 표시된 제목으로 구분할 수 있습니다.
- AIR Introspector를 비 응용 프로그램 샌드박스에서 실행할 때는 [소스] 탭에 응용 프로그램 파일이 표시되지 않습니다.
- AIR Introspector는 자신을 연 샌드박스의 코드만 검사할 수 있습니다.

20장: AIR 응용 프로그램 지역화

Adobe AIR 1.1 이상

Adobe® AIR®에서는 여러 언어를 지원합니다.

ActionScript 3.0 및 Flex 프레임워크의 내용을 지역화하는 방법에 대한 개요는 ActionScript 3.0 개발자 안내서에서 "응용 프로그램 지역화"를 참조하십시오.

AIR에서 지원되는 언어

AIR 1.1 릴리스에는 다음 언어에 대한 AIR 응용 프로그램 지역화 지원이 포함되었습니다.

- 중국어 간체
- 중국어 번체
- 프랑스어
- 독일어
- 이탈리아어
- 일본어
- 한국어
- 포르투갈어(브라질)
- 러시아어
- 스페인어

AIR 1.5 릴리스에는 다음 언어에 대한 지원이 추가되었습니다.

- 체코어
- 네덜란드어
- 폴란드어
- 스웨덴어
- 터키어

기타 도움말 항목

Building multilingual Flex applications on Adobe AIR

Building a multilingual HTML-based application

AIR 응용 프로그램 설치 프로그램에서 응용 프로그램 이름 및 설명 지역화

Adobe AIR 1.1 이상

응용 프로그램 설명자 파일의 name 및 description 요소에 여러 언어를 지정할 수 있습니다. 예를 들어, 다음에서는 3개 언어(영어, 프랑스어 및 독일어)로 응용 프로그램 이름을 지정합니다.

```
<name>
    <text xml:lang="en">Sample 1.0</text>
    <text xml:lang="fr">Échantillon 1.0</text>
    <text xml:lang="de">Stichprobe 1.0</text>
</name>
```

각 text 요소에 대한 xml:lang 특성은 RFC4646(http://www.ietf.org/rfc/rfc4646.txt)에 정의된 언어 코드를 지정합니다.

name 요소는 AIR 설치 프로그램에서 표시하는 응용 프로그램 이름을 정의합니다. AIR 설치 프로그램에서는 운영 체제 설정에 정의된 사용자 인터페이스 언어에 가장 잘 맞는 지역화된 값을 사용합니다.

마찬가지로 응용 프로그램 설명자 파일의 description 요소에도 여러 언어 버전을 지정할 수 있습니다. 이 요소에서는 AIR 설치 프로그램에서 표시하는 설명 텍스트를 정의합니다.

이러한 설정은 AIR 설치 프로그램에서 사용할 수 있는 언어에만 적용되며, 설치되어 실행 중인 응용 프로그램에 사용할 수 있는 로캘을 정의하지는 않습니다. AIR 응용 프로그램에서는 AIR 설치 프로그램에 사용할 수 있는 언어뿐 아니라 여러 언어를 지원하는 사용자 인터페이스를 제공할 수 있습니다.

자세한 내용은 187페이지의 “AIR 응용 프로그램 설명자 요소”를 참조하십시오.

기타 도움말 항목

Building multilingual Flex applications on Adobe AIR

Building a multilingual HTML-based application

AIR HTML 지역화 프레임워크를 사용하여 HTML 내용 지역화

Adobe AIR 1.1 이상

AIR 1.1 SDK에는 HTML 지역화 프레임워크가 포함되어 있습니다. AIRLocalizer.js JavaScript file 파일에서 이 프레임워크를 정의합니다. AIR SDK의 프레임워크 디렉토리에는 AIRLocalizer.js 파일이 포함되어 있습니다. 이 파일에는 여러 지역화된 버전을 지원하는 응용 프로그램을 만들 때 도움이 되는 기능을 제공하는 air.Localizer 클래스가 포함되어 있습니다.

AIR HTML 지역화 프레임워크 코드 로드

지역화 프레임워크를 사용하려면 AIRLocalizer.js 파일을 프로젝트에 복사합니다. 그런 다음 스크립트 태그를 사용하여 이 파일을 응용 프로그램의 주 HTML 파일에 포함합니다.

```
<script src="AIRLocalizer.js" type="text/javascript" charset="utf-8"></script>
```

이후 JavaScript에서는 다음과 같이 air.Localizer.localizer 객체를 호출할 수 있습니다.

```
<script>
    var localizer = air.Localizer.localizer;
</script>
```

air.Localizer.localizer 객체는 지역화된 리소스를 사용 및 관리하기 위한 메서드와 속성을 정의하는 단일 객체입니다. Localizer 클래스에는 다음과 같은 메서드가 포함되어 있습니다.

메서드	설명
getFile()	특정 로캘에 대해 지정된 리소스 번들의 텍스트를 가져옵니다. 274페이지의 “특정 로캘의 리소스 가져오기”를 참조하십시오.
getLocaleChain()	로캘 체인의 언어를 반환합니다. 273페이지의 “로캘 체인 정의”를 참조하십시오.
getResourceBundle()	번들 키 및 해당 값을 객체로 반환합니다. 274페이지의 “특정 로캘의 리소스 가져오기”를 참조하십시오.

메서드	설명
getString()	리소스에 대해 정의된 문자열을 가져옵니다. 274페이지의 "특정 로캘의 리소스 가져오기"를 참조하십시오.
setBundlesDirectory()	번들 디렉토리 위치를 설정합니다. 272페이지의 "AIR HTML Localizer 설정 사용자 정의"를 참조하십시오.
setLocalAttributePrefix()	HTML DOM 요소에 사용된 localizer 특성에서 사용하는 접두어를 설정합니다. 272페이지의 "AIR HTML Localizer 설정 사용자 정의"를 참조하십시오.
setLocaleChain()	로캘 체인의 언어 순서를 설정합니다. 273페이지의 "로캘 체인 정의"를 참조하십시오.
sortLanguagesByPreference()	운영 체제 설정의 로캘 순서를 기준으로 로캘 체인의 로캘을 정렬합니다. 273페이지의 "로캘 체인 정의"를 참조하십시오.
update()	현재 로캘 체인의 지역화된 문자열을 사용하여 HTML DOM(또는 DOM 요소)을 업데이트합니다. 로캘 체인에 대한 자세한 내용은 270페이지의 "로캘 체인 관리"를 참조하십시오. update() 메서드에 대한 자세한 내용은 271페이지의 "현재 로캘을 사용하도록 DOM 요소 업데이트"를 참조하십시오.

Localizer 클래스에는 다음과 같은 정적 속성이 포함됩니다.

속성	설명
localizer	응용 프로그램의 단일 Localizer 객체에 대한 참조를 반환합니다.
ultimateFallbackLocale	응용 프로그램에서 사용자 환경 설정을 지원하지 않는 경우 사용되는 로캘입니다. 273페이지의 "로캘 체인 정의"를 참조하십시오.

지원되는 언어 지정

응용 프로그램 설명자 파일에서 <supportedLanguages> 요소를 사용하여 응용 프로그램에서 지원되는 언어를 식별합니다. 이 요소는 iOS, Mac 전용 런타임 및 Android 응용 프로그램에서만 사용되고 다른 모든 응용 프로그램 유형에서는 무시됩니다.

<supportedLanguages> 요소를 지정하지 않을 경우 기본적으로 패키지 프로그램은 응용 프로그램 유형에 따라 다음 작업을 수행합니다.

- iOS - AIR 런타임이 지원하는 모든 언어가 iOS App Store에서 응용 프로그램의 지원되는 언어로 나열됩니다.
- Mac 전용 런타임 - 전용 번들과 함께 패키지화된 응용 프로그램에 지역화 정보가 없습니다.
- Android - 응용 프로그램 번들에 AIR 런타임이 지원하는 모든 언어에 대한 리소스가 있습니다.

자세한 내용은 215페이지의 "supportedLanguages"를 참조하십시오.

리소스 번들 정의

HTML 지역화 프레임워크는 **지역화** 파일에서 지역화된 버전의 문자열을 읽습니다. 지역화 파일은 텍스트 파일로 직렬화된 키 기반 값의 컬렉션입니다. 지역화 파일은 **번들**이라고도 합니다.

응용 프로그램 프로젝트 디렉토리의 하위 디렉토리를 locale이라는 이름으로 만듭니다. 다른 이름을 사용할 수도 있습니다. 272페이지의 "AIR HTML Localizer 설정 사용자 정의"를 참조하십시오. 이 디렉토리에는 지역화 파일이 포함됩니다. 이 디렉토리를 **번들 디렉토리**라고 합니다.

응용 프로그램에서 지원하는 각 로캘에 대해 번들 디렉토리의 하위 디렉토리를 만듭니다. 각 하위 디렉토리의 이름을 로캘 코드와 같게 지정합니다. 예를 들어 프랑스어 디렉토리는 "fr"로, 영어 디렉토리는 "en"으로 이름을 지정합니다. 밑줄(_) 문자를 사용하여 언어 및 국가 코드가 있는 로캘을 정의할 수 있습니다. 예를 들어 영어(미국) 디렉토리는 "en_us"로 이름을 지정합니다. 또한 "en-us"와 밑줄 대신 하이픈을 사용할 수도 있습니다. HTML 지역화 프레임워크에서는 이 둘을 모두 지원합니다.

로캘 하위 디렉토리에 원하는 수의 리소스 파일을 추가할 수 있습니다. 일반적으로 각 언어에 대해 지역화 파일을 만들어 해당 응용 프로그램의 디렉토리에 저장합니다. HTML 지역화 프레임워크에는 파일의 내용을 읽는 데 사용할 수 있는 getFile() 메서드가 포함되어 있습니다(274페이지의 "특정 로캘의 리소스 가져오기" 참조).

.properties 파일 확장명을 갖는 파일을 지역화 속성 파일이라고 합니다. 이 파일을 사용하여 로캘의 키-값 쌍을 정의할 수 있습니다. 속성 파일에서는 각 행에 하나의 문자열을 정의합니다. 예를 들어 다음은 문자열 값 "Hello in English."(greeting이라는 키에 대해)를 정의합니다.

```
greeting=Hello in English.
```

다음 텍스트가 포함된 속성 파일은 6개의 키-값 쌍을 정의합니다.

```
title=Sample Application
greeting=Hello in English.
exitMessage=Thank you for using the application.
color1=Red
color2=Green
color3=Blue
```

이 예제에서는 en 디렉토리에 저장될 영어 버전의 속성 파일을 보여 줍니다.

이 속성 파일의 프랑스어 버전은 fr 디렉토리에 저장됩니다.

```
title=Application Example
greeting=Bonjour en français.
exitMessage=Merci d'avoir utilisé cette application.
color1=Rouge
color2=Vert
color3=Bleu
```

서로 다른 정보 유형에 대해 여러 리소스 파일을 정의할 수 있습니다. 예를 들어 legal.properties 파일에는 저작권 정보와 같은 공통 조항 법률 텍스트가 포함될 수 있습니다. 또한 여러 응용 프로그램에서 해당 리소스를 다시 사용할 수 있습니다. 마찬가지로 사용자 인터페이스의 서로 다른 부분에 대해 지역화된 내용을 정의하는 별도의 파일을 정의할 수 있습니다.

여러 언어를 지원하려면 이러한 파일에 UTF-8 인코딩을 사용합니다.

로캘 체인 관리

응용 프로그램에서 AIRLocalizer.js 파일을 로드할 때 응용 프로그램에 정의된 로캘을 검사합니다. 이러한 로캘은 번들 디렉토리의 하위 디렉토리에 해당합니다(269페이지의 “리소스 번들 정의” 참조). 이러한 사용 가능한 로캘 목록을 로캘 체인이라고 합니다. AIRLocalizer.js 파일에서는 운영 체제 설정에 정의된 기본 순서에 따라 로캘 체인을 자동으로 정렬합니다. Capabilities.languages 속성에서는 운영 체제 사용자 인터페이스 언어를 기본 순서로 나열합니다.

따라서 응용 프로그램에서 "en", "en_US" 및 "en_UK" 로캘에 대한 리소스를 정의하면 AIR HTML Localizer 프레임워크는 로캘 체인을 적절하게 정렬합니다. "en"을 기본 로캘로 보고하는 시스템에서 응용 프로그램이 시작되는 경우 로캘 체인은 ["en", "en_US", "en_UK"]로 정렬됩니다. 이 경우 응용 프로그램에서는 먼저 "en" 번들의 리소스를 찾은 다음 "en_US" 번들의 리소스를 찾습니다.

그러나 시스템에서 "en-US"를 기본 로캘로 보고하는 경우에는 ["en_US", "en", en_UK"]로 정렬됩니다. 이 경우 응용 프로그램에서는 먼저 "en_US" 번들에서, 그 다음 "en" 번들에서 리소스를 찾습니다.

기본적으로 응용 프로그램에서는 로캘 체인의 첫 번째 로캘을 사용할 기본 로캘로 정의합니다. 응용 프로그램을 처음 실행할 때 사용자에게 로캘을 선택하도록 요청할 수 있습니다. 그런 다음 선택한 내용을 환경 설정 파일에 저장하고 이후 응용 프로그램을 시작할 때 해당 로캘을 사용할 수 있습니다.

응용 프로그램에서는 로캘 체인에 있는 모든 로캘의 리소스 문자열을 사용할 수 있습니다. 특정 로캘에 리소스 문자열이 정의되어 있지 않으면 응용 프로그램은 로캘 체인에 정의된 다른 로캘에서 일치하는 다음 리소스 문자열을 사용합니다.

Localizer 객체의 setLocaleChain() 메서드를 호출하여 로캘 체인을 사용자 정의할 수 있습니다. 273페이지의 “로캘 체인 정의”를 참조하십시오.

지역화된 내용으로 DOM 요소 업데이트

응용 프로그램의 요소에서 지역화 속성 파일을 키 값을 참조할 수 있습니다. 예를 들어 다음 예제의 title 요소는 local_innerHTML 특성을 지정합니다. 지역화 프레임워크에서는 이 특성을 사용하여 지역화된 값을 조회합니다. 기본적으로 프레임워크에서는 "local_"로 시작하는 특성 이름을 찾습니다. 그리고 "local_" 다음의 텍스트와 일치하는 이름의 특성을 업데이트합니다. 이 경우 프레임워크에서는 title 요소의 innerHTML 특성을 설정합니다. innerHTML 특성은 기본 속성 파일(default.properties)의 mainWindowTitle 키에 정의된 값을 사용합니다.

```
<title local_innerHTML="default.mainWindowTitle"/>
```

현재 로캘에 일치하는 값이 정의되어 있지 않으면 localizer 프레임워크는 나머지 로캘 체인을 검색합니다. 로캘 체인에서 값이 정의된 다음 로캘을 사용합니다.

다음 예제에서 p 요소의 텍스트(innerHTML 특성)는 기본 속성 파일에 정의된 greeting 키 값을 사용합니다.

```
<p local_innerHTML="default.greeting" />
```

다음 예제에서 input 요소의 값 특성 및 표시된 텍스트는 기본 속성 파일에 정의된 btnBlue 키의 값을 사용합니다.

```
<input type="button" local_value="default.btnBlue" />
```

현재 로캘 체인에 정의된 문자열을 사용하도록 HTML DOM을 업데이트하려면 Localizer 객체의 update() 메서드를 호출합니다. update() 메서드를 호출하면 Localizer 객체가 DOM을 파싱하고 지역화("local_...") 특성을 찾는 경우 조작을 적용합니다.

```
air.Localizer.localizer.update();
```

특성(예: "innerHTML")과 해당하는 지역화 특성(예: "local_innerHTML") 모두에 대해 값을 정의할 수 있습니다. 이 경우 지역화 프레임워크는 지역화 체인에서 일치하는 값을 찾는 경우에만 특성 값을 덮어씁니다. 예를 들어 다음 요소는 value 및 local_value 특성을 모두 정의합니다.

```
<input type="text" value="Blue" local_value="default.btnBlue"/>
```

특정 DOM 요소만 업데이트할 수도 있습니다. 다음 단원 271페이지의 “현재 로캘을 사용하도록 DOM 요소 업데이트”를 참조하십시오.

기본적으로 AIR HTML Localizer에서는 "local_"을 요소에 대한 지역화 설정을 정의하는 특성의 접두어로 사용합니다. 예를 들어 기본적으로 local_innerHTML 특성은 요소의 innerHTML 값에 사용되는 번들 및 리소스 이름을 정의합니다. 또한 기본적으로 local_value 특성은 요소의 value 특성에 사용되는 번들 및 리소스 이름을 정의합니다. "local_"이 아닌 다른 특성 접두어를 사용하도록 Localizer를 구성할 수 있습니다. 272페이지의 “AIR HTML Localizer 설정 사용자 정의”를 참조하십시오.

현재 로캘을 사용하도록 DOM 요소 업데이트

Localizer 객체가 HTML DOM을 업데이트하면 표시된 요소는 현재 로캘 체인에 정의된 문자열을 기준으로 특성 값을 사용하게 됩니다. HTML localizer가 HTML DOM을 업데이트하게 하려면 Localizer 객체의 update() 메서드를 호출합니다.

```
air.Localizer.localizer.update();
```

지정한 DOM 요소만 업데이트하려면 해당 요소를 update() 메서드에 매개 변수로 전달합니다. update() 메서드에는 선택 사항인 한 개의 메서드 parentNode만 있습니다. parentNode 매개 변수는 지정하는 경우 지역화할 DOM 요소를 정의합니다. update() 메서드를 호출하고 parentNode 매개 변수를 지정하면 지역화 특성을 지정하는 모든 자식 요소에 대한 지역화된 값이 설정됩니다.

예를 들어, 다음 div 요소를 살펴봅니다.

```
<div id="colorsDiv">
    <h1 local_innerHTML="default.lblColors" ></h1>
    <p><input type="button" local_value="default.btnBlue" /></p>
    <p><input type="button" local_value="default.btnRed" /></p>
    <p><input type="button" local_value="default.btnGreen" /></p>
</div>
```

현재 로캘 체인에 정의된 지역화된 문자열을 사용하도록 이 요소를 업데이트하려면 다음 JavaScript 코드를 사용합니다.

```
var divElement = window.document.getElementById("colorsDiv");
air.Localizer.localizer.update(divElement);
```

로캘 체인에 키 값이 없는 경우 지역화 프레임워크에서는 특성 값을 "local_" 특성의 값으로 설정합니다. 예를 들어 앞의 예제에서 지역화 프레임워크가 로캘 체인의 모든 default.properties 파일에서 lblColors 키 값을 찾을 수 없다고 가정합니다. 이 경우 프레임워크는 "default.lblColors"를 innerHTML 값으로 사용합니다. 이 값을 사용하면 개발자에게 누락된 리소스를 표시할 수 있습니다.

update() 메서드는 로캘 체인에서 리소스를 찾을 수 없는 경우 resourceNotFound 이벤트를 전달합니다. air.Localizer.RESOURCE_NOT_FOUND 상수는 문자열 "resourceNotFound"를 정의합니다. 이 이벤트에는 bundleName, resourceName 및 locale의 세 가지 속성이 있습니다. bundleName 속성은 리소스가 없는 번들의 이름입니다. resourceName 속성은 리소스가 없는 번들의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다.

update() 메서드는 지정한 번들을 찾을 수 없는 경우 bundleNotFound 이벤트를 전달합니다. air.Localizer.BUNDLE_NOT_FOUND 상수는 문자열 "bundleNotFound"를 정의합니다. 이 이벤트에는 bundleName 및 locale의 두 가지 속성이 있습니다. bundleName 속성은 리소스가 없는 번들의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다.

update() 메서드는 비동기적으로 작동하며 resourceNotFound 및 bundleNotFound 이벤트를 비동기적으로 전달합니다. 다음 코드에서는 resourceNotFound 및 bundleNotFound 이벤트에 대한 이벤트 리스너를 설정합니다.

```
air.Localizer.localizer.addEventListener(air.Localizer.RESOURCE_NOT_FOUND, rnfHandler);
air.Localizer.localizer.addEventListener(air.Localizer.BUNDLE_NOT_FOUND, rnfHandler);
air.Localizer.localizer.update();
function rnfHandler(event)
{
    alert(event.bundleName + ": " + event.resourceName + ":." + event.locale);
}
function bnfHandler(event)
{
    alert(event.bundleName + ":." + event.locale);
}
```

AIR HTML Localizer 설정 사용자 정의

Localizer 객체의 setBundlesDirectory() 메서드를 사용하면 번들 디렉토리 경로를 사용자 정의할 수 있습니다. Localizer 객체의 setLocalAttributePrefix() 메서드를 사용하면 번들 디렉토리 경로를 사용자 정의하고 Localizer에서 사용하는 특성 값을 사용자 정의할 수 있습니다.

기본 번들 디렉토리는 응용 프로그램 디렉토리의 로캘 하위 디렉토리로 정의됩니다. Localizer 객체의 setBundlesDirectory() 메서드를 호출하여 다른 디렉토리를 지정할 수 있습니다. 이 메서드에서는 하나의 매개 변수 path를 사용하며 이 매개 변수는 원하는 번들 디렉토리의 경로를 문자열로 지정합니다. path 매개 변수 값은 다음 중 하나가 될 수 있습니다.

- 응용 프로그램 디렉토리에 상대적인 경로를 정의하는 문자열(예: "locales")
- app, app-storage 또는 file URL 스킴을 사용하는 유효한 URL을 정의하는 문자열(예: "app://languages"). http URL 스킴은 사용하지 마십시오.
- File 객체

URL 및 디렉토리 경로에 대한 자세한 내용은 다음을 참조하십시오.

- File 객체의 경로(ActionScript 개발자용)
- Paths of File objects(HTML 개발자용)

예를 들어 다음 코드에서는 번들 디렉토리를 응용 프로그램 디렉토리가 아닌 응용 프로그램 저장소 디렉토리의 languages 하위 디렉토리로 설정합니다.

```
air.Localizer.localizer.setBundlesDirectory("languages");
```

유효한 경로를 path 매개 변수로 전달합니다. 그렇지 않으면 메서드에서 BundlePathNotFoundError 예외가 발생합니다. 이 오류의 name 속성은 "BundlePathNotFoundError"이고 message 속성은 잘못된 경로를 지정합니다.

기본적으로 AIR HTML Localizer에서는 "local_"을 요소에 대한 지역화 설정을 정의하는 특성의 접두어로 사용합니다. 예를 들어 local_innerHTML 특성은 다음 input 요소의 innerHTML 값에 사용되는 번들 및 리소스 이름을 정의합니다.

```
<p local_innerHTML="default.greeting" />
```

Localizer 객체의 setLocalAttributePrefix() 메서드를 사용하면 "local_"이 아닌 특성 접두어를 사용할 수 있습니다. 이 정적 메서드에서는 특성 접두어로 사용할 문자열인 하나의 매개 변수를 사용합니다. 예를 들어 다음 코드에서는 "loc_"를 특성 접두어로 사용하도록 지역화 프레임워크를 설정합니다.

```
air.Localizer.localizer.setLocalAttributePrefix("loc_");
```

지역화 프레임워크에서 사용하는 특성 접두어를 사용자 정의할 수 있습니다. 기본값("local_")이 코드에 사용되는 다른 특성의 이름과 충돌하는 경우 접두어를 사용자 정의할 수 있습니다. 이 메서드를 호출할 때는 HTML 특성에 유효한 문자를 사용해야 합니다. 예를 들어 값에 공백 문자가 포함될 수 없습니다.

HTML 요소에서 지역화 특성 사용에 대한 자세한 내용은 271페이지의 “지역화된 내용으로 DOM 요소 업데이트”를 참조하십시오.

번들 디렉토리 및 특성 접두어 설정은 응용 프로그램 세션이 바뀌면 유지되지 않습니다. 사용자 정의 번들 디렉토리 또는 특성 접두어 설정을 사용하려면 응용 프로그램이 시작할 때마다 이를 설정해야 합니다.

로캘 체인 정의

기본적으로 AIRLocalizer.js 코드를 로드할 때 기본 로캘 체인이 설정됩니다. 로캘은 번들 디렉토리에서 사용할 수 있고 운영 체제 언어 설정에서 이 로캘 체인을 정의합니다. 자세한 내용은 270페이지의 “로캘 체인 관리”를 참조하십시오.

Localizer 객체의 정적 setLocaleChain() 메서드를 호출하여 로캘 체인을 수정할 수 있습니다. 예를 들어 특정 언어를 선호하는 경우 이 메서드를 호출할 수 있습니다. setLocaleChain() 메서드는 하나의 매개 변수 chain을 사용합니다. 이 매개 변수는 ["fr_FR","fr","fr_CA"]과 같은 로캘 배열입니다. 배열의 로캘 순서에 따라 후속 작업에서 프레임워크가 리소스를 찾는 순서가 결정됩니다. 체인의 첫 번째 로캘에서 리소스를 찾지 못하면 계속 다른 로캘의 리소스를 찾습니다. chain 인수가 없거나 배열이 아니거나 빈 배열이면 함수가 실패하고 IllegalArgumentsError 예외가 발생합니다.

Localizer 객체의 정적 getLocaleChain() 메서드는 현재 로캘 체인의 로캘을 나열하는 배열을 반환합니다.

다음 코드에서는 현재 로캘 체인을 읽고 체인의 머리 부분에 두 개의 프랑스어 로캘을 추가합니다.

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
```

setLocaleChain() 메서드는 로캘 체인을 업데이트할 때 "change" 이벤트를 전달합니다. air.Localizer.LOCALE_CHANGE 상수는 문자열 "change"를 정의합니다. 이 이벤트에서는 새 로캘 체인의 로캘 코드 배열인 localeChain이라는 하나의 속성을 사용합니다. 다음 코드에서는 이 이벤트에 대한 이벤트 리스너를 설정합니다.

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
localizer.addEventListener(air.Localizer.LOCALE_CHANGE, changeHandler);
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
function changeHandler(event)
{
    alert(event.localeChain);
}
```

정적 air.Localizer.ultimateFallbackLocale 속성은 응용 프로그램이 사용자 환경 설정을 지원하지 않는 경우 사용되는 로캘을 나타냅니다. 기본값은 "en"입니다. 다음 코드와 같이 이를 다른 로캘로 설정할 수 있습니다.

```
air.Localizer.ultimateFallbackLocale = "fr";
```

특정 로캘의 리소스 가져오기

Localizer 객체의 getString() 메서드는 특정 로캘의 리소스에 대해 정의된 문자열을 반환합니다. 이 메서드를 호출할 때 locale 값을 지정할 필요가 없습니다. 이 경우 메서드에서는 전체 로캘 체인을 확인하고 지정된 리소스 이름을 제공하는 첫 번째 로캘의 문자열을 반환합니다. 이 메서드의 매개 변수는 다음과 같습니다.

매개 변수	설명
bundleName	리소스를 포함하는 번들입니다. .properties 확장명을 제외한 속성 파일의 파일 이름입니다. 예를 들어 이 매개 변수를 "alerts"로 설정하면 Localizer 코드에서는 alerts.properties라는 지역화 파일을 확인합니다.
resourceName	리소스 이름입니다.
templateArgs	선택 사항입니다. 대체 문자열에서 번호 지정된 태그를 대체할 문자열 배열입니다. 예를 들어 templateArgs 매개 변수가 ["Raúl", "4"]이고 일치하는 리소스 문자열이 "Hello, {0}. You have {1} new messages."인 함수를 호출한다고 가정합니다. 이 경우 함수는 "Hello, Raúl. You have 4 new messages."를 반환합니다. 이 설정을 무시하려면 null 값을 전달합니다.
locale	선택 사항입니다. 사용할 로캘 코드입니다(예: "en", "en_us" 또는 "fr"). 로캘을 지정하고 일치하는 값이 없으면 메서드에서는 로캘 체인의 다른 로캘에서 값을 검색하지 않습니다. 로캘 코드를 지정하지 않으면 함수에서는 지정된 리소스 이름에 대한 값을 제공하는 로캘 체인의 첫 번째 로컬에 있는 문자열을 반환합니다.

지역화 프레임워크에서 표시된 HTML DOM 특성을 업데이트할 수 있습니다. 그러나 지역화된 문자열을 다른 방법으로 사용할 수 있습니다. 예를 들어 문자열을 동적으로 생성된 HTML에서 사용하거나 함수 호출의 매개 변수로 사용할 수 있습니다. 예를 들어 다음 코드에서는 fr_FR 로캘의 기본 속성 파일에서 error114 리소스에 정의된 문자열을 사용하여 alert() 함수를 호출합니다.

```
alert(air.Localizer.localizer.getString("default", "error114", null, "fr_FR"));
```

getString() 메서드는 지정된 번들에서 리소스를 찾을 수 없는 경우 resourceNotFound 이벤트를 전달합니다. air.Localizer.RESOURCE_NOT_FOUND 상수는 문자열 "resourceNotFound"를 정의합니다. 이 이벤트에는 bundleName, resourceName 및 locale의 세 가지 속성이 있습니다. bundleName 속성은 리소스가 없는 번들의 이름입니다. resourceName 속성은 리소스가 없는 번들의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다.

getString() 메서드는 지정된 번들을 찾을 수 없는 경우 bundleNotFound 이벤트를 전달합니다. air.Localizer.BUNDLE_NOT_FOUND 상수는 문자열 "bundleNotFound"를 정의합니다. 이 이벤트에는 bundleName 및 locale의 두 가지 속성이 있습니다. bundleName 속성은 리소스가 없는 번들의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다.

getString() 메서드는 비동기적으로 작동하며 resourceNotFound 및 bundleNotFound 이벤트를 비동기적으로 전달합니다. 다음 코드에서는 resourceNotFound 및 bundleNotFound 이벤트에 대한 이벤트 리스너를 설정합니다.

```
air.Localizerlocalizer.addEventListener(air.Localizer.RESOURCE_NOT_FOUND, rnfHandler);
air.Localizerlocalizer.addEventListener(air.Localizer.BUNDLE_NOT_FOUND, bnfHandler);
var str = air.Localizer.localizer.getString("default", "error114", null, "fr_FR");
function rnfHandler(event)
{
    alert(event.bundleName + ": " + event.resourceName + ":." + event.locale);
}
function bnfHandler(event)
{
    alert(event.bundleName + ":." + event.locale);
}
```

Localizer 객체의 getResourceBundle() 메서드는 제공된 로캘에 대해 지정된 번들을 반환합니다. 이 메서드의 반환 값은 번들의 키와 일치하는 속성이 있는 객체입니다. 응용 프로그램에서 지정된 번들을 찾을 수 없는 경우 이 메서드는 null을 반환합니다.

이 메서드는 locale 및 bundleName이라는 두 가지 매개 변수를 사용합니다.

매개 변수	설명
locale	로캘(예: "fr")입니다.
bundleName	번들 이름입니다.

예를 들어 다음 코드에서는 document.write() 메서드를 호출하여 fr 로캘의 기본 번들을 로드합니다. 그런 다음 document.write() 메서드를 호출하여 해당 번들에 있는 str1 및 str2 키의 값을 작성합니다.

```
var aboutWin = window.open();
var bundle = localizer.getResourceBundle("fr", "default");
aboutWin.document.write(bundle.str1);
aboutWin.document.write("<br/>");
aboutWin.document.write(bundle.str2);
aboutWin.document.write("<br/>");
```

getResourceBundle() 메서드는 지정된 번들을 찾을 수 없는 경우 bundleNotFound 이벤트를 전달합니다. air.Localizer.BUNDLE_NOT_FOUND 상수는 문자열 "bundleNotFound"를 정의합니다. 이 이벤트에는 bundleName 및 locale의 두 가지 속성이 있습니다. bundleName 속성은 리소스가 없는 번들의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다.

Localizer 객체의 getFile() 메서드는 지정된 로캘에 대해 번들의 내용을 문자열로 반환합니다. 번들 파일은 UTF-8 파일로 읽습니다. 이 메서드의 매개 변수는 다음과 같습니다.

매개 변수	설명
resourceFileName	리소스 파일의 파일 이름입니다(예: "about.html").

templateArgs	선택 사항입니다. 대체 문자열에서 번호 지정된 태그를 대체할 문자열 배열입니다. 예를 들어 templateArgs 매개 변수가 ["Raúl", "4"]이고 일치하는 리소스 파일에 두 줄이 포함된 함수를 호출한다고 가정합니다.
	
	`<html>`
	`<body>Hello, {0}. You have {1} new messages.</body>`
	`</html>`
	
	이 경우 함수는 다음과 같이 두 줄이 포함된 문자열을 반환합니다.
	
	`<html>`
	`<body>Hello, Raúl. You have 4 new messages. </body>`
	`</html>`

locale	사용할 로캘 코드입니다(예: "en_GB"). 로캘을 지정하고 일치하는 파일이 없으면 메서드에서는 로캘 체인의 다른 로캘에서 검색을 계속하지 않습니다. 로캘 코드를 지정하지 **않으면** 함수에서는 resourceFileName과 일치하는 파일이 있는 로캘 체인의 첫 번째 로캘에 있는 텍스트를 반환합니다.

예를 들어 다음 코드에서는 fr 로캘의 about.html 파일의 내용을 사용하여 document.write() 메서드를 호출합니다.

```
var aboutWin = window.open();
var aboutHtml = localizer.getFile("about.html", null, "fr");
aboutWin.document.close();
aboutWin.document.write(aboutHtml);
```

getFile() 메서드는 로캘 체인에서 리소스를 찾을 수 없는 경우 fileNotFound 이벤트를 전달합니다. air.Localizer.FILE_NOT_FOUND 상수는 문자열 "resourceNotFound"를 정의합니다. getFile() 메서드는 비동기적으로 작동하며 fileNotFound 이벤트를 비동기적으로 전달합니다. 이 이벤트에는 fileName 및 locale의 두 가지 속성이 있습니다. fileName 속성은 찾지 못한 파일의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다. 다음 코드에서는 이 이벤트에 대한 이벤트 리스너를 설정합니다.

```
air.Localizer.localizer.addEventListener(air.Localizer.FILE_NOT_FOUND, fnfHandler);
air.Localizer.localizer.getFile("missing.html", null, "fr");
function fnfHandler(event)
{
    alert(event.fileName + ": " + event.locale);
}
```

기타 도움말 항목

Building a multilingual HTML-based application

21장: path 환경 변수

AIR SDK에는 명령줄 또는 터미널에서 실행할 수 있는 몇 가지 프로그램이 있습니다. SDK bin 디렉토리의 경로가 path 환경 변수에 포함되어 있으면 이러한 프로그램을 더 편리하게 실행할 수 있는 경우가 많습니다.

여기에 소개된 정보는 Windows, Mac 및 Linux에서 경로를 설정하는 방법에 대해 다루며, 간편한 가이드로 사용해야 합니다. 하지만 컴퓨터 구성은 매우 다양하기 때문에 일부 시스템에서는 이 절차가 맞지 않을 수 있습니다. 그럴 경우에는 운영 체제 설명서 또는 인터넷에서 필요한 정보를 찾을 수 있습니다.

Bash 셸을 사용하여 Linux 및 Mac OS에서 PATH 설정

터미널 윈도우에서 명령을 입력하면 입력된 내용을 읽고 이에 맞게 응답하려고 시도하는 프로그램인 셸이 먼저 파일 시스템에서 명령 프로그램을 찾아야 합니다. 셸은 $PATH라는 환경 변수에 저장된 디렉토리 목록에서 명령을 찾습니다. path에 현재 나열된 항목을 확인하려면 다음을 입력하십시오.

```
echo $PATH
```

그러면 다음과 같이 콜론으로 구분된 디렉토리 목록이 반환됩니다.

```
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin
```

목표는 셸이 ADT 및 ADL 도구를 찾을 수 있도록 AIR SDK bin 디렉토리의 경로를 이 목록에 추가하는 것입니다. /Users/fred/SDKs/AIR에 AIR SDK를 넣었다고 가정하면 다음 명령에 따라 필요한 디렉토리가 path에 추가됩니다.

```
export PATH=$PATH:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

참고: path에 공백 문자가 포함되어 있으면 다음과 같이 백슬래시로 이스케이프 처리하십시오.

```
/Users/fred\ jones/SDKs/AIR\ 2.5\ SDK/bin
```

echo 명령을 다시 사용하여 제대로 되었는지 확인할 수 있습니다.

```
echo $PATH
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

지금까지는 제대로 되었습니다. 이제 다음 명령을 입력하고 희망적인 응답을 얻을 수 있을 것입니다.

```
adt -version
```

$PATH 변수를 제대로 수정했다면 이 명령이 ADT의 버전을 보고할 것입니다.

하지만 그래도 한 가지 문제가 있습니다. 다음에 새 터미널 윈도우를 실행할 때 path의 새 항목이 더 이상 그곳에 없는 것을 발견하게 될 것입니다. 새 터미널을 시작할 때마다 path를 설정하는 명령을 실행해야 합니다.

이 문제에 대한 공통된 해결책은 셸이 사용하는 시작 스크립트 중 하나에 명령을 추가하는 것입니다. Mac OS에서는 ~/username 디렉토리에 .bash_profile이라는 파일을 만들 수 있습니다. 그러면 새 터미널 윈도우를 열 때마다 이 파일이 실행됩니다. Ubuntu에서는 새 터미널 윈도우를 실행할 때 .bashrc라는 시작 스크립트가 실행됩니다. 다른 Linux 배포 및 셸 프로그램에도 비슷한 규약이 있습니다.

셸 시작 스크립트에 명령을 추가하려면

1. 홈 디렉토리를 변경합니다.

   ```
   cd
   ```

2. 셸 구성 프로파일을 만들고(필요한 경우) "cat >>"를 사용하여 입력되는 텍스트를 파일 끝으로 리디렉션합니다. 운영 체제 및 셸에 알맞은 파일을 사용합니다. 예를 들어 Mac OS에는 .bash_profile을, Ubuntu에는 .bashrc를 사용할 수 있습니다.

   ```
   cat >> .bash_profile
   ```

3 파일에 추가할 텍스트를 입력합니다.

```
export PATH=$PATH:/Users/cward/SDKs/android/tools:/Users/cward/SDKs/AIR/bin
```

4 키보드에서 CTRL-SHIFT-D를 눌러 텍스트 리디렉션을 종료합니다.

5 파일을 표시하여 모든 것이 제대로 되었는지 확인합니다.

```
cat .bash_profile
```

6 새 터미널 윈도우를 열어서 path를 확인합니다.

```
echo $PATH
```

추가된 경로가 나열되어야 합니다.

나중에 다른 디렉토리에서 한 SDK의 새 버전을 만들 경우에는 구성 파일에서 path 명령을 업데이트해야 합니다. 이렇게 하지 않으면 셸이 계속 이전 버전을 사용합니다.

Windows에서 path 설정

Windows에서 명령 윈도우를 열 경우 해당 윈도우는 시스템 속성에 정의된 전역 환경 변수를 상속받습니다. 중요한 변수 중 하나가 path입니다. path는 실행할 프로그램의 이름을 입력할 때 명령 프로그램이 검색하는 디렉토리의 목록입니다. 명령 윈도우를 사용할 때 path에 현재 포함되어 있는 내용을 보려면 다음을 입력하면 됩니다.

```
set path
```

그러면 다음과 같이 세미콜론으로 구분된 디렉토리 목록이 표시됩니다.

```
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
```

목표는 명령 프로그램이 ADT 및 ADL 도구를 찾을 수 있도록 AIR SDK bin 디렉토리의 경로를 이 목록에 추가하는 것입니다. C:\SDKs\AIR에 AIR SDK를 넣었다고 가정하면 다음 절차를 통해 올바른 path 항목을 추가할 수 있습니다.

1 제어판에서 또는 [내 컴퓨터]를 마우스 오른쪽 버튼으로 클릭하고 메뉴에서 [속성]을 선택하여 [시스템 속성] 대화 상자를 엽니다.

2 [고급] 탭 아래에서 [환경 변수] 버튼을 클릭합니다.

3 [환경 변수] 대화 상자의 [시스템 변수] 섹션에서 Path 항목을 선택합니다.

4 [편집]을 클릭합니다.

5 [변수 값] 필드에서 텍스트의 끝으로 스크롤합니다.

6 현재 값의 맨 끝에 다음 텍스트를 입력합니다.

```
;C:\SDKs\AIR\bin
```

7 모든 대화 상자에서 [확인]을 클릭하여 path를 저장합니다.

명령 윈도우가 열려 있으면 환경이 업데이트되지 않습니다. 새 명령 윈도우를 열고 다음 명령을 입력하여 경로가 올바르게 설정되었는지 확인합니다.

```
adt -version
```

나중에 AIR SDK의 위치를 변경하거나 새 버전을 추가할 경우 path 변수를 꼭 업데이트해야 합니다.