建立 ADOBE® AIR® 應用程式

上次更新 **2017/5/31**

法律聲明

內容

第 1 章 關於 Adobe AIR

Adobe® AIR® 是一個多重作業系統的多螢幕執行階段，可讓您利用所擁有的網頁開發技巧建立多樣化網際網路應用程式 (RIA)，並部署至桌面與行動裝置。您可以使用 Adobe® Flex 與 Adobe® Flash® (SWF 類型)，透過 ActionScript 3.0 來建立桌上型、電視版和行動裝置版 AIR 應用程式。您也可以使用 HTML、JavaScript® 以及 Ajax (HTML 類型) 來建立桌上型 AIR 應用程式。

您可以在 Adobe AIR Developer Connection 找到有關 Adobe AIR 快速入門和使用的詳細資訊：http://www.adobe.com/devnet/air/。

AIR 可讓您在熟悉的環境中工作，並使用您認為最得心應手的工具和方式。透過對 Flash、Flex、HTML、JavaScript 和 Ajax 的支援，建置符合您需求的最佳體驗。

例如，您可以使用下列其中一項或多項技巧來開發應用程式：

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

使用者與 AIR 應用程式互動的方式和他們與原生應用程式互動的方式相同。只要在使用者的電腦或裝置中安裝一次執行階段，之後就可以像其他桌上型應用程式一樣安裝並執行多個 AIR 應用程式 (在 iOS 上，不會安裝另一個 AIR 執行階段，每個 iOS AIR 應用程式都是獨立的應用程式)。

執行階段會提供用於部署應用程式的一致性跨作業系統平台和架構，因此藉由確保能跨電腦提供一致的功能和互動，便不需執行跨瀏覽器測試。您可以將目標鎖定為執行階段，而非針對特定作業系統進行開發，如此具有下列優點：

- 為 AIR 開發的應用程式可以跨多個作業系統執行，您不需再多費工夫。在 AIR 支援的所有作業系統中，執行階段可以確保產生一致且可預期的表現與互動。
- 藉由利用所擁有的網頁技巧和設計模式，您可以更快速地建置應用程式。您可以將網頁架構應用程式擴充至桌上型電腦，而不需學習傳統的電腦部署技巧或複雜的原生程式碼。
- 相較之下，部署應用程式比使用較低階的語言 (例如 C 和 C++) 更為容易。您不需要管理每個作業系統專屬的複雜低階 API。

為 AIR 開發應用程式時，您可以善用豐富的架構和 API 資源：

- 執行階段和 AIR 架構所提供的 AIR 專屬 API
- 在 SWF 檔和 Flex 架構 (以及其它以 ActionScript 為基礎的元件庫和架構) 中使用的 ActionScript API
- HTML、CSS 和 JavaScript
- 大部分 Ajax 架構
- Adobe AIR 原生擴充功能的 ActionScript API，可讓您存取以原生程式碼進行程式設計的特定平台功能。原生擴充功能也可讓您存取舊版原生程式碼，以及提供更高效能的原生程式碼。

AIR 大幅改變了應用程式的建立、部署和體驗方式。您可以獲得更多具有創意的控制方式，並且將您的 Flash、Flex、HTML 和 Ajax 類型應用程式擴充至桌上型電腦、行動裝置和電視。

如需有關每個新 AIR 更新內容的詳細資訊，請參閱 Adobe AIR 版本說明 (http://www.adobe.com/go/learn_air_relnotes_tw)。

第 2 章 Adobe AIR 安裝

The Adobe® AIR® 執行階段可讓您執行 AIR 應用程式。您可以透過下列方式安裝執行階段：

- 另外安裝執行階段 (與 AIR 應用程式分開安裝)
- 透過網頁安裝 「標誌」首次安裝 AIR 應用程式 (將提示您同時安裝執行階段)
- 建立可安裝應用程式與執行階段的自訂安裝程式。您必須取得 Adobe 的許可才能以這種方式散佈 AIR 執行階段。您可在 Adobe runtime licensing 頁面上請求許可。請注意，Adobe 不提供建立這種安裝程式的工具。但是，有許多可用的協力廠商安裝程式工具套件。
- 藉由安裝組合 AIR 為固定執行階段的 AIR 應用程式。僅有組合應用程式才能使用固定執行階段。這不是用於執行其他 AIR 應用程式。組合執行階段是 Mac 和 Windows 上的選項。在 iOS 上，所有應用程式都有組合的執行階段。從 AIR 3.7 開始，所有 Android 應用程式預設都會包含組合的執行階段 (儘管您可以選擇使用個別執行階段)。
- 設定 AIR 開發環境，例如 AIR SDK、Adobe® Flash® Builder™ 或 Adobe Flex® SDK (其中包含 AIR 命令列開發工具). 包括在 SDK 中的執行階段只能用於應用程式除錯，不能用於執行已安裝的 AIR 應用程式。

下列網站詳細說明安裝與執行 AIR 應用程式的系統需求：Adobe AIR：系統需求 (http://www.adobe.com/tw/products/air/systemreqs/)。

執行階段安裝程式與 AIR 應用程式安裝程式在安裝、更新或移除 AIR 應用程式或 AIR 執行階段本身時，都會建立記錄檔。您可以查閱這些記錄檔，判斷任何安裝問題的發生原因。請參閱安裝記錄。

安裝 Adobe AIR

若要安裝或更新執行階段，使用者必須具有電腦的系統管理權限。

在 **Windows** 電腦中安裝執行階段

1. 從 http://get.adobe.com/tw/air 下載執行階段安裝檔案。
2. 按兩下這個執行階段安裝檔案。
3. 在安裝視窗中，遵循提示以完成安裝程序。

在 **Mac** 電腦中安裝執行階段

1. 從 http://get.adobe.com/tw/air 下載執行階段安裝檔案。
2. 按兩下這個執行階段安裝檔案。
3. 在安裝視窗中，遵循提示以完成安裝程序。
4. 如果安裝程式出現 「驗證」視窗，請輸入 Mac OS 的使用者名稱和密碼。

在 **Linux** 電腦中安裝執行階段

備註：目前，Linux 都不支援 AIR 2.7 與更新的版本。部署至 Linux 的 AIR 應用程式應該持續使用 AIR 2.6 SDK。

使用二進位安裝程式：

1. 從 http://kb2.adobe.com/tw/cps/853/cpsid_85304.html 中尋找並下載安裝二進位檔案。
2. 設定檔案的權限，使安裝應用程式得以執行。從命令列可以透過下列步驟設定檔案權限：

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

某些版本的 Linux 可以讓您在 「屬性」對話方塊 (透過快顯選單開啟) 中設定檔案權限。

3 從命令列執行安裝程式，或按兩下執行階段安裝檔案。

4 在安裝視窗中，遵循提示以完成安裝程序。

Adobe AIR 會安裝為原生套件。換句話說，以 RPM 為基礎的散發版本將以 RPM 安裝，Debian 散發版本則會以 DEB 安裝。目前 AIR 不支援任何其他套件格式。

使用套件安裝程式：

1 從 http://kb2.adobe.com/tw/cps/853/cpsid_85304.html 找出 AIR 套件檔案。視您系統支援的套件格式而定，下載 RPM 或 Debian 套件。

2 如有需要，按兩下 AIR 套件檔案來安裝套件。

您也可以從命令列安裝：

a 在 Debian 系統上：

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

b 在以 RPM 為基礎的系統上：

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

或者，如果您要更新現有的版本 (AIR 1.5.3 或更新的版本)：

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

若要安裝 AIR 2 與 AIR 應用程式，您必須具有電腦的系統管理員權限。

Adobe AIR 會安裝至下列位置：/opt/Adobe AIR/Versions/1.0

AIR 會註冊 mime-type "application/vnd.adobe.air-application-installer-package+zip"，這表示該 .air 檔案為此 mime-type，因此會在 AIR 執行階段註冊。

在 **Android** 裝置上安裝執行階段

您可以安裝從 Android Market 取得的 AIR 執行階段最新版本。

您可以從網頁連結或是使用 ADT -installRuntime 命令，安裝 AIR 執行階段的開發版本。一次只能安裝一個 AIR 執行階段版本，您無法同時安裝發行版本與開發版本。

如需詳細資訊，請參閱 第 151 頁 「ADT installRuntime 命令」。

在 **iOS** 裝置上安裝執行階段

所需的 AIR 執行階段程式碼隨附於為 iPhone、iTouch 和 iPad 裝置建立的每個應用程式中。您不需要個別安裝執行階段組件。

更多說明主題

第 61 頁 「AIR for iOS」

移除 Adobe AIR

一旦執行階段安裝完畢，您可以遵循下列步驟來加以移除。

在 **Windows** 電腦中移除執行階段

1 在 Windows 的 「開始」選單中，選取 「設定 > 控制台」。

2 開啟「程式集」、「程式和功能」或「新增或移除程式」控制台 (視您執行的 Windows 版本而定)。

3 選取「Adobe AIR」以移除執行階段。

4 按一下「變更 / 移除」按鈕。

在 **Mac** 電腦中移除執行階段

- 按兩下位於 /Applications/Utilities 資料夾中的「Adobe AIR Uninstaller」。

在 **Linux** 電腦中移除執行階段

執行下列其中一項操作：

- 在「應用程式」選單中選取「Adobe AIR Uninstaller」命令。
- 以 -uninstall 選項執行 AIR 二進位安裝程式
- 使用套件管理員移除 AIR 套件 (adobeair 和 adobecerts)。

從 **Android** 裝置移除執行階段

1 在裝置上開啟「設定」應用程式。

2 輕點「應用程式 > 管理應用程式」下的 Adobe AIR 項目。

3 輕點「解除安裝」按鈕。

您也可以使用 ADT -uninstallRuntime 命令。如需詳細資訊，請參閱 第 152 頁「ADT uninstallRuntime 命令」。

解決組合的執行階段

若要改善固定組合的執行階段，您必須移除用來安裝它的應用程式。請注意，固定執行階段僅可用於執行安裝應用程式。

安裝與執行 AIR 樣本應用程式

若要安裝或更新 AIR 應用程式，使用者必須具有電腦的系統管理權限。

有些樣本應用程式可用來示範 AIR 的功能。您可以遵循下列步驟，存取並安裝這些應用程式：

1 下載並執行「AIR 樣本應用程式」，其中會提供已編譯的應用程式和原始碼。

2 若要下載與執行樣本應用程式，請按一下樣本應用程式的「立即安裝」按鈕。您會收到提示，詢問是否要安裝與執行應用程式。

3 如果您想要先下載樣本應用程式並於稍後執行，請選取下載連結。您隨時可以透過下列方式執行 AIR 應用程式：

- 在 Windows 中，按兩下桌面上的應用程式圖示，或者從 Windows 的「開始」選單選取應用程式圖示。
- 在 Mac OS 中，按兩下應用程式圖示。這個應用程式圖示預設會安裝在使用者目錄的 Applications 資料夾中 (例如，Macintosh HD/Users/JoeUser/Applications/)。

備註：如需與這些指示相關的更新內容，請查看 AIR 版本說明 (位於 http://www.adobe.com/go/learn_air_relnotes_tw)。

Adobe AIR 更新

Adobe 會定期更新 Adobe AIR 來添加新的功能或非嚴重性問題的修正檔案。「自動通知和更新」功能會在有可用的 Adobe AIR 更新版本時，讓 Adobe 自動通知使用者。

Adobe AIR 更新不但可確保 Adobe AIR 維持正常運作，而且通常包含重要的安全性變更。只要有可用的最新 Adobe AIR 版本，Adobe 便會建議使用者立即更新至最新的版本，特別是當新版涉及安全性更新的時候。

根據預設，啟動 AIR 應用程式時，執行階段會檢查是否有可用的更新。如果上次檢查更新的時間為兩個星期之前，便會執行這項檢查。如果有可用的更新，AIR 將在背景下載更新。

使用者可以使用 AIR SettingsManager 應用程式來停用自動更新功能。您可以從 http://airdownload.adobe.com/air/applications/SettingsManager/SettingsManager.air 下載 AIR SettingsManager 應用程式。

Adobe AIR 的正常安裝程序包括連線至 http://airinstall.adobe.com 以傳送與安裝環境相關的基本資訊，例如作業系統版本和語言。每次安裝時只會傳送一次此資訊，如此可允許 Adobe 確認安裝是否成功。不會收集或傳送個人身分資訊。

更新固定執行階段

如果您以固定執行階段組合包來散佈應用程式，則不會自動更新固定執行階段。為了使用者的安全性考量，您必須監控 Adobe 發佈的更新，並且在發佈相關安全性變更時，以新版執行階段更新您的應用程式。

第 3 章 使用 AIR API

Adobe® AIR® 包括在 Adobe® Flash® Player 中執行的 SWF 內容所無法使用的功能。

ActionScript 3.0 開發人員
Adobe AIR API 記載在下列兩本書籍中：

- ActionScript 3.0 開發人員指南
- 適用於 Adobe Flash Platform 的 ActionScript 3.0 參考

HTML 開發人員
如果您要建立 HTML 類型的 AIR 應用程式，下列兩本書籍將說明透過 AIRAliases.js 檔案 (請參閱 Accessing AIR API classes from JavaScript) 以 JavaScript 提供給您的 API：

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

AIR 專屬 ActionScript 3.0 類別

下表包含 Adobe AIR 專屬的執行階段類別。它們無法提供給在瀏覽器 Adobe® Flash® Player 中執行的 SWF 內容使用。

HTML 開發人員
透過 AIRAliases.js 檔案以 JavaScript 提供給您的類別會列在 Adobe AIR API Reference for HTML Developers 中。

類別	**ActionScript 3.0** 套件	新增於 **AIR** 版本
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
CompressionAlgorithm	flash.utils	1.0

類別	ActionScript 3.0 套件	新增於 AIR 版本
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 和更早版本；已丟棄，從 3.7 開始
GameInputDevice	flash.ui	3.0
GameInputEvent	flash.ui	3.0
GameInputFinger	flash.ui	3.6 和更早版本；已丟棄，從 3.7 開始
GameInputHand	flash.ui	3.6 和更早版本；已丟棄，從 3.7 開始
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

類別	ActionScript 3.0 套件	新增於 AIR 版本
HTMLUncaughtScriptExceptionEvent	flash.events	1.0
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

類別	ActionScript 3.0 套件	新增於 AIR 版本
NativeWindowType	flash.display	1.0
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

類別	ActionScript 3.0 套件	新增於 AIR 版本
SQLSchema	flash.data	1.0
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

具備 AIR 專屬功能的 Flash Player 類別

下列類別適用於瀏覽器中執行的 SWF 內容，但 AIR 還另外提供其它屬性或方法：

套件	類別	屬性、方法或事件	新增於 AIR 版本
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 事件	2.0
		orientationChanging 事件	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

套件	類別	屬性、方法或事件	新增於 AIR 版本
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

套件	類別	屬性、方法或事件	新增於 AIR 版本
flash.net	FileReference	extension	1.0
		httpResponseStatus 事件	1.0
		uploadUnencoded()	1.0
	NetStream	drmAuthenticate 事件	1.0
		onDRMContentData 事件	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 事件	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

套件	類別	屬性、方法或事件	新增於 AIR 版本
flash.system	Capabilities	languages	1.1
	LoaderContext	allowLoadBytesCodeExecution	1.0
	Security	APPLICATION	1.0
flash.ui	KeyLocation	D_PAD	2.5

這些新屬性和方法大多僅適用於 AIR 應用程式安全執行程序中的內容。不過，URLRequest 類別的新成員也適用於在其它安全執行程序中執行的內容。

ByteArray.compress() 方法和 ByteArray.uncompress() 方法都新增了 algorithm 參數，讓您能夠選擇使用 deflate 壓縮或 zlib 壓縮。此參數只適用於 AIR 中所執行的內容。

AIR 專屬的 Flex 組件

開發 Adobe AIR 內容時，您可以使用下列 Adobe® Flex™ MX 組件：

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

此外，Flex 4 包含下列 Spark AIR 組件：

- Window
- WindowedApplication

如需 AIR Flex 元件的詳細資訊，請參閱 Using the Flex AIR components。

第 4 章 AIR 開發適用的 Adobe Flash Platform 工具

您可以利用下列 Adobe Flash Platform 開發工具來開發 AIR 應用程式。

ActionScript 3.0 (Flash 與 Flex) 開發人員：

- Adobe Flash Professional (請參閱 AIR 的發佈功能)
- Adobe Flex 3.x 和 4.x SDK (請參閱第 16 頁 「設定 Flex SDK」和 第 142 頁 「AIR Developer Tool (ADT)」)
- Adobe Flash Builder (請參閱使用 Flash Builder 開發 AIR 應用程式)

HTML 與 Ajax 開發人員：

- Adobe AIR SDK (請參閱第 15 頁 「安裝 AIR SDK」和 第 142 頁 「AIR Developer Tool (ADT)」)
- Adobe Dreamweaver CS3、CS4、CS5 (請參閱適用於 Dreamweaver 的 AIR 擴充功能)

安裝 AIR SDK

Adobe AIR SDK 包含下列可用來啟動和封裝應用程式的命令列工具：

AIR Debug Launcher (ADL) 可讓您不需安裝 AIR 應用程式即可執行。請參閱 第 137 頁 「AIR Debug Launcher (ADL)」。

AIR Development Tool (ADT) 將 AIR 應用程式封裝成可散佈的安裝套件。請參閱 第 142 頁 「AIR Developer Tool (ADT)」。

AIR 命令列工具要求您的電腦必須安裝 Java。您可以從 JRE 或 JDK (1.5 版或更新版本) 使用 Java 虛擬機器。Java JRE 與 Java JDK 可從 http://java.sun.com/ 取得。

至少需要 2GB 的電腦記憶體才能使用 ADT 工具。

備註：使用者則不需 Java 也可執行 AIR 應用程式。

如需使用 AIR SDK 建立 AIR 應用程式的概觀，請參閱第 29 頁 「使用 AIR SDK 建立您的第一個 HTML 類型 AIR 應用程式」。

下載和安裝 AIR SDK

您可以遵循下列指示，下載並安裝 AIR SDK：

在 **Windows** 中安裝 **AIR SDK**

- 下載 AIR SDK 安裝檔案。
- AIR SDK 以標準存檔檔案的形式散佈。若要安裝 AIR，請將 SDK 的內容解壓縮到電腦的資料夾 (例如：C:\Program Files\Adobe\AIRSDK 或 C:\AIRSDK)。
- AIR SDK 的 bin 資料夾中包含 ADL 與 ADT 工具；請將此資料夾的路徑加到 PATH 環境變數中。

在 **Mac OS X** 中安裝 **AIR SDK**

- 下載 AIR SDK 安裝檔案。

- AIR SDK 以標準存檔檔案的形式散佈。若要安裝 AIR，請將 SDK 的內容解壓縮到電腦的資料夾 (例如：/Users/< 使用者名稱 >/Applications/AIRSDK)。
- AIR SDK 的 bin 資料夾中包含 ADL 與 ADT 工具：請將此資料夾的路徑加到 PATH 環境變數中。

在 **Linux** 中安裝 **AIR SDK**

- SDK 可提供 tbz2 格式。
- 若要安裝 SDK，請建立要在其中解壓縮 SDK 的資料夾，然後使用下列命令：tar -jxvf <AIR-SDK.tbz2 的路徑 >

如需使用 AIR SDK 工具的快速入門詳細資訊，請參閱 「使用命令列工具建立 AIR 應用程式」。

AIR SDK 的內容

下表說明 AIR SDK 中所包含檔案的用途：

SDK 資料夾	檔案 / 工具說明
bin	AIR Debug Launcher (ADL) 可讓您不需要先封裝並安裝 AIR 應用程式，即可執行 AIR 應用程式。如需有關使用此工具的詳細資訊，請參閱 第 137 頁 「AIR Debug Launcher (ADL)」。 AIR Developer Tool (ADT) 可將應用程式封裝成可供散佈的 AIR 檔案。如需有關使用此工具的詳細資訊，請參閱 第 142 頁 「AIR Developer Tool (ADT)」。
frameworks	libs 目錄包含要用於 AIR 應用程式的程式碼元件庫。 專案目錄包含編譯的 SWF 與 SWC 元件庫的程式碼。
include	所附的目錄包含撰寫原生擴充功能的 C 語言標題檔案。
install	install 目錄包含 Android 裝置適用的 Windows USB 驅動程式 (這些是 Google 提供給 Android SDK 的驅動程式)。
lib	包含 AIR SDK 工具的支援程式碼。
runtimes	適用於桌上型電腦與行動裝置的 AIR 執行階段。 ADL 使用的桌上型執行階段，可在封裝或安裝 AIR 應用程式之前啟動 AIR 應用程式。 您可以在 Android 裝置或模擬器上安裝 Android 的 AIR 執行階段 (APK 套件) 以進行開發和測試。裝置和模擬器分別使用不同的 APK 套件 (適用於 Android 的公開 AIR 執行階段可從 Android Market 取得)。
samples	此資料夾包含應用程式描述器檔案樣本、隱藏安裝功能 (badge.swf) 的樣本以及預設的 AIR 應用程式圖示。
templates	descriptor-template.xml - 每個 AIR 應用程式所需之應用程式描述器檔案的樣本。如需有關應用程式描述器檔案的詳細說明，請參閱 第 173 頁 「AIR 應用程式描述器檔案」。 在此資料夾中也可以找到適用於每個 AIR 發行版本之應用程式描述器的 XML 結構檔案。

設定 Flex SDK

若要利用 Adobe® Flex™ 來開發 Adobe® AIR® 應用程式，您有下列選擇：

- 您可以下載和安裝 Adobe® Flash® Builder™，此程式提供的整合工具可建立 Adobe AIR 專案，並測試、除錯和封裝 AIR 應用程式。請參閱第 18 頁 「在 Flash Builder 中建立第一個桌上型 Flex AIR 應用程式」。
- 您可以使用偏好的文字編輯器與命令列工具，下載 Adobe® Flex™ SDK 和開發 Flex AIR 應用程式。

如需使用 Flex SDK 建立 AIR 應用程式的概觀，請參閱第 32 頁 「使用 Flex SDK 建立您的第一個桌上型 AIR 應用程式」。

安裝 Flex SDK

您的電腦必須安裝有 Java，才能使用命令列工具建立 AIR 應用程式。您可以從 JRE 或 JDK (1.5 版或更新版本) 使用 Java 虛擬機器。Java JRE 與 JDK 可從 http://java.sun.com/ 取得。

備註：使用者則不需 Java 也可執行 AIR 應用程式。

Flex SDK 提供的 AIR API 與命令列工具，可協助您封裝、編譯以及除錯 AIR 應用程式。

1 如果您尚未這樣做，請從 http://opensource.adobe.com/wiki/display/flexsdk/Downloads 下載 Flex SDK。

2 將 SDK 的內容放入資料夾 (例如 Flex SDK)。

3 將 AIR SDK 的內容複製到 Flex SDK 的檔案中。

   備註：在 Mac 電腦上，請確定您複製或取代　SDK 資料夾中的個別檔案，而不是整個目錄。根據預設，將 Mac 上的目錄複製到相同名稱的目錄會移除目標目錄中的現有檔案，並不會合併兩個目錄的內容。您可在終端視窗中使用 ditto 命令來將 AIR SDK 合併至 Flex SDK：ditto air_sdk_folder flex_sdk_folder

4 命令列 AIR 公用程式位於 bin 資料夾。

設定外部 SDK

開發 Android 與 iOS 的應用程式需要您從平台製造商下載佈建檔案、SDK 或其他開發工具。

如需有關下載和安裝 Android SDK 的詳細資訊，請參閱 Android 開發人員：安裝 SDK。從 AIR 2.6 開始，不再需要下載 Android SDK。AIR SDK 現在包括安裝和啟動 APK　套件所需的基本組件。但 Android SDK 對於各種開發工作而言仍然非常有用，包括建立和執行軟體模擬器和擷取裝置螢幕快照。

開發 iOS 不需要外部 SDK。不過，需要特殊的憑證與佈建描述檔。如需詳細資訊，請參閱從 Apple 取得開發人員檔案。

第 5 章 建立您的第一個 AIR 應用程式

在 Flash Builder 中建立第一個桌上型 Flex AIR 應用程式

若要親身體驗 Adobe® AIR® 的運作方式，請使用這些指示在 Adobe® Flash® Builder 建立和封裝一個簡單的 SWF 檔案類型 AIR "Hello World" 應用程式。

如果您尚未下載和安裝 Flash Builder，請進行下載和安裝。另外，請移至下列網址下載並安裝最新版的 Adobe AIR：http://www.adobe.com/tw/products/air。

建立 AIR 專案

Flash Builder 包含開發和封裝 AIR 應用程式的工具。

在 Flex Builder 或 Flex Builder 中開始建立 AIR 應用程式的方式和建立其他 Flex 類型應用程式專案相同，都是從定義新專案開始。

1 開啟 Flash Builder。

2 選取 「檔案 > 新增 > Flex 專案」。

3 輸入 AIRHelloWorld 當做專案名稱。

4 在 Flex 中，AIR 應用程式被視為一種應用程式類型。您有兩個類型選項：

- 在 Adobe® Flash® Player 中執行的網頁應用程式
- 在 Adobe AIR 中執行的桌上型應用程式

請選取 「桌上型」做為應用程式類型

5 按一下 「完成」來建立專案。

AIR 專案最初由兩個檔案組成：主要 MXML 檔案和應用程式 XML 檔案 (稱為應用程式描述器檔案)。後者檔案會指定應用程式屬性。

如需詳細資訊，請參閱使用 Flash Builder 開發 AIR 應用程式。

撰寫 AIR 應用程式程式碼

若要撰寫 “Hello World” 應用程式程式碼，請編輯開啟於編輯器中的應用程式 MXML 檔案 (AIRHelloWorld.mxml) (如果檔案未開啟，請使用 「專案導覽器」開啟檔案)。

桌上型的 Flex AIR 應用程式包含在 MXML WindowedApplication 標記中。MXML WindowedApplication 標籤會建立簡單的視窗，其中包含基本的視窗控制項，例如標題列和關閉按鈕。

1 在 WindowedApplication 組件加入 title 特質並指派值 "Hello World"：

```
<?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 在應用程式中加入 Label 組件 (放置在 WindowedApplication 標籤內)。將 Label 組件的 text 屬性設成 "Hello AIR"，並將佈局限制設成保持在中央，如以下所示：

```
<?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 在 WindowedApplication 標籤開頭的後方以及您剛輸入之 Label 組件標籤的前方加入下列樣式區塊：

```
<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>
```

這些樣式設定會套用至整個應用程式，並將視窗背景顯示成稍微透明的灰色。

應用程式的程式碼現在看起來如下：

```
<?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>
```

接下來，您會變更應用程式描述器中的某些設定，讓應用程式呈現透明效果：

1 在「Flex 導覽器」窗格，從專案的來源目錄中找出應用程式描述器檔案。如果您將專案命名為 AIRHelloWorld，則這個檔案的名稱是 AIRHelloWorld-app.xml。

2 按兩下應用程式描述器檔案，以 Flash Builder 進行編輯。

3 在 XML 程式碼中，找出將 systemChrome 和 transparent 屬性 (屬於 initialWindow 屬性) 設成註解的列。移除註解標記。(移除 "<!--" 和 "-->" 註解分隔符號。)

4 將 systemChrome 屬性的文字值設成 none，如以下所示：

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

5 將 transparent 屬性的文字值設成 true，如以下所示：

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

6 儲存檔案。

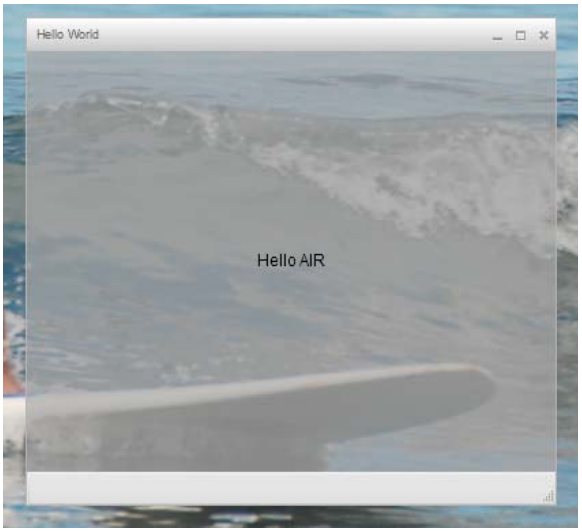
測試 AIR 應用程式

若要測試您撰寫的應用程式程式碼，請以除錯模式執行程式碼。

1 按一下主工具列中的「除錯」按鈕 。

您也可以選取「執行 > 除錯 > AIRHelloWorld」命令。

產生的 AIR 應用程式看起來應該如下列範例所示：



2 使用 Label 控制項的 horizontalCenter 和 verticalCenter 屬性，將文字放置在視窗的中央。移動或調整視窗大小的方式，與其他任何桌面應用程式相同。

備註：如果無法編譯應用程式，請修正不小心輸入的任何語法或拼字錯誤。錯誤和警告會顯示在 Flash Builder 的「問題」檢視中。

封裝、簽署和執行 AIR 應用程式

您現在可以將 "Hello World" 應用程式封裝成 AIR 檔進行散佈。AIR 檔是包含應用程式檔案的存檔檔案，即專案 bin 資料夾中的所有檔案。在這個簡單範例中，那些檔案包括 SWF 和應用程式的 XML 檔案。您將 AIR 套件散佈給使用者，使用者使用該套件來安裝應用程式。這個程序中有一個必要步驟，就是進行數位簽署。

1 確定應用程式沒有編譯錯誤而且正常執行。

2 選取「專案 > 匯出發行組建」。

3 確定列出的專案和應用程式是 AIRHelloWorld 專案和 AIRHelloWorld.mxml 應用程式。

4 選取「匯出為簽署的 AIR 套件」選項，然後按「下一步」。

5 如果您已經有數位憑證，請按一下「瀏覽」來尋找並選取數位憑證。

6 如果您需要建立新的自我簽署數位憑證，請選取「建立」。

7 輸入必要資訊，然後按一下「確定」。

8 按一下「完成」來產生 AIR 套件，名稱是 AIRHelloWorld.air。

您現在可以從 Flash Builder 中的「專案導覽器」，或從檔案系統按兩下 AIR 檔案來安裝並執行應用程式。

使用 Flash Professional 建立您的第一個桌上型 AIR 應用程式

若要親身體驗 Adobe® AIR® 的運作方式，請依照本主題的指示，使用 Adobe® Flash® Professional 建立與封裝簡單的 "Hello World" AIR 應用程式。

如果您未曾嘗試過，請下載和安裝 Adobe AIR，網址為：www.adobe.com/go/air_tw。

使用 Flash 建立 Hello World 應用程式

使用 Flash 建立 Adobe AIR 應用程式的方式與建立任何其它 FLA 檔案的方式大致相同。下列程序將引導您使用 Flash Professional 逐步建立簡單的 Hello World 應用程式。

建立 **Hello World** 應用程式

1. 啟動 Flash。
2. 在「歡迎螢幕」上按一下 AIR，以使用 Adobe AIR 發佈設定建立空白 FLA 檔案。
3. 從「工具」面板選取「文字」工具，並在「舞台」正中央建立靜態文字欄位 (預設)。調整文字欄位的寬度，使其足以容納 15 至 20 個字元。
4. 在文字欄位中輸入 “Hello World” 文字。
5. 儲存並命名檔案 (例如 HelloAIR)。

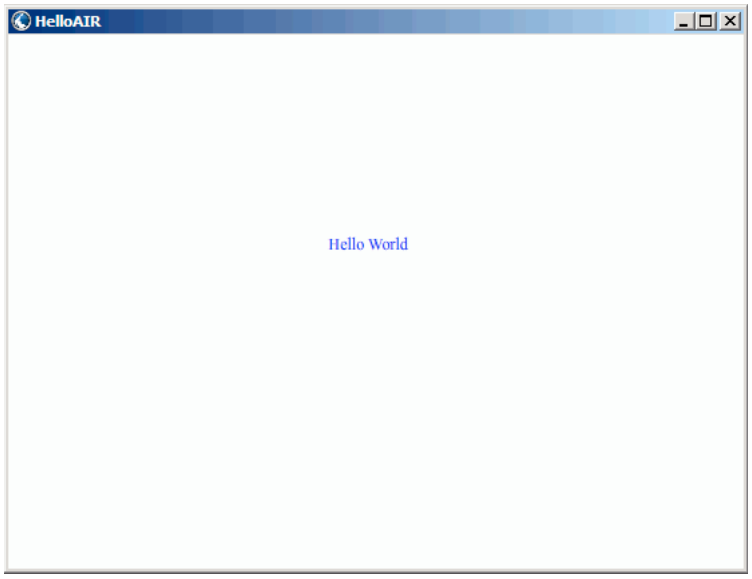
測試應用程式

1. 按 Ctrl+Enter 或選取「控制 > 測試影片 > 測試」，以便在 Adobe AIR 中測試應用程式。
2. 若要使用「影片除錯」功能，請先在應用程式中加入 ActionScript 程式碼。您可以加入如下所示的 trace 陳述式，快速地嘗試除錯：

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

3. 按 Ctrl+Shift+Enter 或選取「除錯 > 影片除錯 > 除錯」，使用「影片除錯」來執行應用程式。

Hello World 應用程式看起來如下圖所示：



封裝應用程式

1 選取「檔案」>「發佈」。

2 使用現有的數位憑證來簽署 Adobe AIR 套件，或使用下列步驟建立自我簽署憑證：

   a 按一下「憑證」欄位旁邊的「新增」按鈕。

   b 完成「發行者名稱」、「組織單位」、「組織名稱」、「電子郵件」、「國家 / 地區」、「密碼」和「確認密碼」等項目。

   c 指定憑證的類型。憑證「類型」選項表示安全性等級：1024-RSA 使用 1024 位元金鑰 (較不安全)，而 2048-RSA 則使用 2048 位元金鑰 (較安全)。

   d 完成「另存新檔」項目，或按一下「瀏覽 ...」按鈕並瀏覽到資料夾位置，以將這些資訊儲存成憑證檔案 (例如，C:/Temp/mycert.pfx)。當您完成之後，按一下「確定」。

   e Flash 會返回「數位簽名」對話方塊。您所建立的自我簽署數位憑證，其路徑與檔案名稱將出現在「憑證」文字方塊中。如果沒有，請輸入路徑與檔案名稱，或按一下「瀏覽」按鈕以找出並選取該檔案。

   f 在「數位簽名」對話方塊的「密碼」文字欄位中，輸入您在步驟 b 指定的密碼。如需簽署 Adobe AIR 應用程式的詳細資訊，請參閱第 161 頁「為 AIR 檔加上數位簽名」。

3 若要建立應用程式和安裝程式檔案，請按一下「發佈」按鈕 (在 Flash CS4 與 CS5 中，按一下「確定」按鈕)。建立 AIR 檔之前，您必須執行「測試影片」或「影片除錯」先建立 SWF 檔和 application.xml 檔。

4 若要安裝應用程式，請按兩下儲存應用程式之資料夾中的 AIR 檔 (application.air)。

5 在「應用程式安裝」對話方塊中，按一下「安裝」按鈕。

6 檢閱「安裝偏好設定」和「位置」設定，並確定已選取「安裝後啟動應用程式」核取方塊，然後按一下「繼續」。

7 出現「安裝已完成」訊息時，按一下「完成」。

在 Flash Professional 中建立您的第一個 AIR for Android 應用程式

若要開發 Android 適用的 AIR 應用程式，您必須下載 Android 的 Flash Professional CS5 擴充功能，網址為：Adobe Labs。

您必須另外從 Android 網站下載和安裝 Android SDK，如需詳細資訊，請參閱：Android 開發人員：安裝 SDK。

建立專案

1 開啟 Flash Professional CS5

2 建立新的 AIR for Android 專案。

Flash Professional 主畫面包括建立 AIR for Android 應用程式的連結。您也可以選取 「檔案 > 新增」，然後選取 AIR for Android 範本。

3 將文件儲存為 HelloWorld.fla

撰寫程式碼

此教學課程不是真的在說明如何撰寫程式碼，主要目的是在說明使用 「文字工具」在舞台上撰寫 "Hello, World!" 的方式。

設定應用程式屬性

1 選取 「檔案 > AIR Android 設定」。

2 在 「一般」標籤中，進行以下設定：

- 輸出檔案：HelloWorld.apk
- 應用程式名稱：HelloWorld
- 應用程式 ID：HelloWorld
- 版本號碼：0.0.1
- 外觀比例：直向

3 在 「部署」標籤中，進行以下設定：

- 憑證：指向有效的 AIR 程式碼簽署憑證。您可以按一下 「建立」按鈕以建立新憑證 (透過 Android Marketplace 部署的 Android 應用程式必須擁有有效期至少到 2033 年的憑證)。在 「密碼」欄位中輸入憑證密碼。
- Android 部署類型：除錯
- 發佈後：兩個選項皆選取
- 輸入 Android SDK tools 子目錄中的 ADB 工具路徑。

4 按一下 「確定」以關閉 Android 設定對話方塊。

在應用程式開發的這個階段，應用程式並不需要圖示或權限。大部分的 AIR for Android 應用程式會要求必須具有某些權限，才能存取受保護的功能。您應該只設定應用程式真正需要的權限，因為如果應用程式要求太多權限，使用者可能會拒絕使用您的應用程式。

5 儲存檔案。

在 **Android** 裝置上封裝和安裝應用程式

1 請確定裝置已啟用 USB 除錯。您可以在 「應用程式 > 開發」下的 「設定」應用程式中，開啟 USB 除錯。

2 使用 USB 纜線將裝置連接到電腦。

3 如果您尚未安裝 AIR 執行階段，請前往 Android Market 並下載 Adobe AIR 以進行安裝 (您也可以使用 第 151 頁「ADT installRuntime 命令」在本機安裝 AIR。適用於 Android 裝置與模擬器的 Android 套件包含在 AIR SDK 中)。

4 選取 「檔案」 >「發佈」。

Flash Professional 會建立 APK 檔案，然後在連接的 Android 裝置上安裝並啟動應用程式。

建立您的第一個 AIR for iOS 應用程式

AIR 2.6 或更新的版本、**iOS 4.2** 或更新的版本

您只能使用 Adobe 工具來撰寫、建立和測試 iOS 應用程式基本功能的程式碼。不過，若要在裝置上安裝 iOS 應用程式並散佈該應用程式，您必須加入 Apple iOS Developer 計劃 (這是付費型服務)。在您加入 iOS Developer 計劃後，即可存取 iOS Provisioning Portal，您可以在此從 Apple 取得下列項目與檔案，這些項目與檔案將用以在裝置上安裝應用程式，以供測試和後續散佈之用。這些項目與檔案包括：

- 開發與發佈憑證
- 應用程式 ID
- 開發與發佈佈建檔案

建立應用程式內容

建立顯示 “Hello world!” 文字的 SWF 檔案您可以使用 Flash Professional、Flash Builder 或其他 IDE 執行此工作。此範例只會使用文字編輯器以及包括在 Flex SDK 中的命令列 SWF 編譯器。

1 在便利的位置中建立目錄以儲存您的應用程式檔案。建立名為 HelloWorld.as 的檔案，並在您偏好的程式碼編輯器中編輯檔案。

2 請加入下列程式碼：

```
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 使用 amxmlc 編譯器編譯類別：

```
amxmlc HelloWorld.as
```

即會在相同的資料夾中建立 SWF 檔案 HelloWorld.swf。

備註：此範例假設您已設定環境 Path 變數來加入含有 amxmlc 的目錄。如需設定路徑的資訊，請參閱 第 263 頁「Path 環境變數」。您也可以輸入 amxmlc 的完整路徑以及此範例中所使用的其他命令列工具。

建立應用程式的圖示圖案及起始螢幕圖案

所有 iOS 應用程式在 iTunes 應用程式的使用者介面及裝置螢幕上都會有自己的圖示。

1 在專案目錄中建立目錄，並命名為 icons。

2 在 icons 目錄中建立 3 個 PNG 檔。將這 3 個檔案分別命名為 Icon_29.png、Icon_57.png 及 Icon_512.png。

3 編輯 PNG 檔，為您的應用程式建立適當的圖案。檔案必須是 29 x 29 像素、57 x 57 像素及 512 x 512 像素。在此測試中，您可以使用實色的矩形做為圖案。

備註：將應用程式送出至 Apple App Store 時，您將使用 512 像素的 JPG 檔 (不是 PNG 檔)。測試應用程式的開發版本時，您便可以使用 PNG 檔。

所有 iPhone 應用程式在載入 iPhone 時都會顯示其起始影像。您會在 PNG 檔定義起始影像：

1 在主要開發目錄中，建立名為 Default.png 的 PNG 檔。(請勿將此檔案放入 icons 子目錄中，務必將檔案命名為大寫 D 的 Default.png)。

2 編輯檔案，讓檔案的寬度為 320 像素，而高度為 480 像素。就現在而言，內容可以是簡單的白色矩形 (我們將在稍後予以變更)。

如需這些圖像的詳細資訊，請參閱第 75 頁 「應用程式圖示」。

建立應用程式描述器檔案

建立應用程式描述器檔案以指定應用程式的基本屬性。您可以使用 IDE (例如 Flash Builder 或文字編輯器) 來完成此工作。

1 在包含 HelloWorld.as 的專案資料夾中，建立名為 HelloWorld-app.xml 的 XML 檔案。在您偏好的 XML 編輯器中編輯此檔案。

2 新增下列 XML：

```
<?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>
```

基於簡潔的考量，此範例只會設定部分可用屬性。

備註：若是使用 AIR 2 或更早版本，您必須使用 <version> 元素而非 <versionNumber> 元素。

3 變更應用程式 ID 以符合在 iOS Provisioning Portal 中指定的應用程式 ID (請勿在 ID 的開頭包括隨機組合包種子部分)。

4 使用 ADL 測試應用程式：

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

ADL 應該會在桌面上開啟顯示下列文字的視窗：Hello World! 如果未開啟，請檢查原始碼與應用程式描述器是否有錯誤。

編譯 IPA 檔

您現在可以使用 ADT 來編譯 IPA 安裝程式檔案。您必須擁有 P12 檔案格式的 Apple 開發人員憑證與私密金鑰，以及 Apple 開發佈建描述檔。

使用下列選項執行 ADT 公用程式，以自己的值取代 keystore、storepass 及 provisioning-profile 值：

```
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
```

(請使用單一命令列；此範例中的斷行符號是為了方便您的閱讀而加入)。

ADT 會在專案目錄中產生 iOS 應用程式安裝程式檔案：HelloWorld.ipa。編譯 IPA 檔需要花費數分鐘的時間。

在裝置上安裝應用程式

安裝 iOS 應用程式以供測試：

1. 開啟 iTunes 應用程式。
2. 如果您尚未這樣做，請將此應用程式的佈建描述檔加到 iTunes 中。在 iTunes 中，選取 「檔案 > 加到資料庫」。然後，選取佈建描述檔 (檔案類型為 mobileprovision)。

   現在，請使用開發佈建描述檔在開發人員裝置上測試應用程式。

   稍後要將應用程式散發至 iTunes Store 時，請使用散發描述檔。若要散發臨時應用程式 (至多個裝置，但不經過 iTunes Store)，請使用臨時佈建描述檔。

   如需有關佈建描述檔的詳細資訊，請參閱 第 57 頁 「iOS 設定」。
3. 在部分 iTunes 版本中，即使您已安裝相同版本的應用程式，新安裝的應用程式並不會取代已安裝的應用程式。在這種情況下，請從裝置及 iTunes 的應用程式清單刪除應用程式。
4. 按兩下應用程式的 IPA 檔。該檔案應該會顯示在 iTunes 的應用程式清單中。
5. 將裝置連接至電腦的 USB 連接埠。
6. 在 iTunes 中，選取裝置的 「應用程式」標籤，並確認在要安裝的應用程式清單中已選取該應用程式。
7. 在 iTunes 應用程式的左側清單選取裝置。然後按一下 「同步」按鈕。完成同步之後，Hello World 應用程式就會顯示在您的 iPhone 中。

如果還是未安裝新的版本，請從裝置及 iTunes 的應用程式清單中刪除該應用程式，然後重做此程序。發生這種情況的原因可能是目前安裝的版本與新安裝的版本使用相同的應用程式 ID 和版本號碼。

編輯起始螢幕圖形

編譯應用程式之前，您可以建立一個 Default.png 檔案 (請參閱第 25 頁 「建立應用程式的圖示圖案及起始螢幕圖案」)。此 PNG 檔會在載入應用程式時，做為啟動影像使用。在 iPhone 測試應用程式時，您會在啟動時看到此空白螢幕。

您應該將此影像變更為與應用程式相符的啟動螢幕 ("Hello World!")：

1. 在裝置上啟動應用程式。首次顯示 “Hello World” 文字時，請按住「主畫面」按鈕 (螢幕下方)。按住「主畫面」按鈕的同時，請按住 「電源 / 睡眠」按鈕 (在 iPhone 的最上方)。這樣會擷取一張螢幕快照並傳送至 「相機膠卷」。

2 藉由從 iPhoto 或其他照片傳送應用程式傳送照片，您可以將影像傳送至您的開發電腦 (在 Mac OS，您也可以使用影像擷取應用程式)。

您也可以將照片透過電子郵件傳送至開發電腦：

- 開啟 「照片」應用程式。
- 開啟 「相機膠卷」。
- 開啟您擷取的螢幕快照影像。
- 點選影像，然後點選左下角的 「下一頁」 (箭號) 按鈕。然後按一下 「透過電子郵件傳送照片」按鈕，將影像傳送給自己。

3 以螢幕擷取影像的 PNG 版本取代 Default.png 檔案 (位於開發目錄中)。

4 重新編譯應用程式 (請參閱第 26 頁 「編譯 IPA 檔」)，並將該應用程式重新安裝到您的裝置。

從現在起，應用程式會於載入時使用新的啟動螢幕。

備註：您可以為 Default.png 檔案建立任何圖案，但外觀比例必須正確 (320 x 480 像素)。不過，最好還是讓 Default.png 影像與應用程式的起始狀態相符。

使用 Dreamweaver 建立您的第一個 HTML 類型 AIR 應用程式

若要親身體驗 Adobe® AIR® 的運作方式，請使用這些指示以及 Adobe® AIR® Extension for Dreamweaver® 來建立和封裝一個簡單的 HTML 類型 AIR "Hello World" 應用程式。

如果您未曾嘗試過，請下載和安裝 Adobe AIR，網址為：www.adobe.com/go/air_tw。

如需適用於 Dreamweaver 的 Adobe AIR 擴充功能安裝指示，請參閱安裝適用於 Dreamweaver 的 AIR 擴充功能。

如需擴充功能的概觀 (包括系統需求)，請參閱適用於 Dreamweaver 的 AIR 擴充功能。

備註：您只能針對 desktop 及 extendedDesktop 描述檔開發 HTML 類型的 AIR 應用程式。不支援行動描述檔。

準備應用程式檔案

您的 Adobe AIR 應用程式必須在 Dreamweaver 網站定義開始頁面 (及其所有相關頁面)：

1 啟動 Dreamweaver，並確定您已經定義一個網站。

2 選取 「檔案 > 新增」，接著在 「頁面類型」欄中選取 HTML，然後在 「佈局」欄中選取 「無」，並按一下 「建立」來開啟新 HTML 頁面。

3 在新頁面中輸入 **Hello World!**

這個範例非常簡單，但您也可以嘗試設定喜好的文字樣式、在頁面中加入更多內容、在開始頁面中加入其他頁面的連結等等。

4 將頁面儲存 (「檔案 > 儲存檔案」) 為 hello_world.html。請確定將檔案儲存在 Dreamweaver 網站中。

如需 Dreamweaver 網站的相關資訊，請參閱 「Dreamweaver 說明」：

建立 Adobe AIR 應用程式

1 請確定您已經在 「Dreamweaver 文件」視窗開啟 hello_world.html 頁面 (請參閱上一節取得建立的指示)。

2 選取 「網站 > AIR 應用程式設定」。

「AIR 應用程式和設定」對話方塊會自動移入大部分的必要設定。不過，您必須自行選取應用程式的初始內容 (或開始頁面) 。

3 按一下 「初始內容」選項旁的 「瀏覽」按鈕，接著瀏覽至您的 hello_world.html 頁面並選取該頁面。

4 按一下 「數位簽名」選項旁的 「設定」按鈕。

數位簽名可以保證應用程式的程式碼在軟體作者建立後未曾變更或損毀，而且所有 Adobe AIR 應用程式都需要數位簽名。

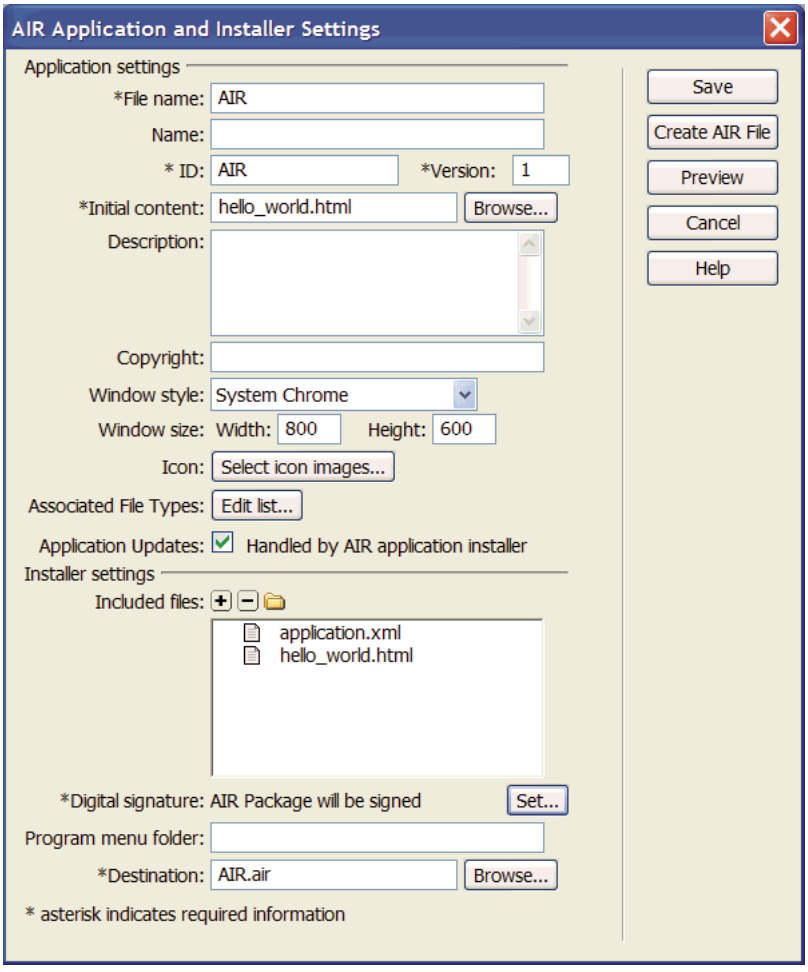
5 在 「數位簽名」對話方塊選取 「使用數位憑證簽署 AIR 套件」，然後按一下 「建立」按鈕。(如果您有數位憑證的存取權，請按一下 「瀏覽」按鈕來選取。)

6 完成 「自我簽署的數位憑證」對話方塊中的必要欄位。您需要輸入您的姓名，輸入密碼並確認無誤，然後再輸入數位憑證檔案的名稱。Dreamweaver 會將數位憑證儲存在您的網站根目錄中。

7 按一下 「確定」回到 「數位簽名」對話方塊。

8 在 「數位簽名」對話方塊，輸入指定的數位憑證的密碼，然後按一下 「確定」。

您完成的 「AIR 應用程式和安裝程式設定」對話方塊可能看起來如下圖：



如需所有對話方塊選項的進一步說明以及其編輯方式，請參閱在 Dreamweaver 中建立 AIR 應用程式。

9 按一下 「建立 AIR 檔」按鈕。

Dreamweaver 隨即建立 Adobe AIR 應用程式檔案，並儲存在您的網站根目錄資料夾中。Dreamweaver 還會建立一個 application.xml 檔案，並儲存在相同位置中。這個檔案提供清單功能，定義應用程式的各種屬性。

將應用程式安裝到桌面

建立應用程式檔案後，可以將它安裝到桌面任何位置。

1 將 Adobe AIR 應用程式檔案從 Dreamweaver 網站移至您的桌面或其他桌面上。

這個步驟是選擇性的。如果您喜歡，也可以從 Dreamweaver 網站將新應用程式實際安裝在電腦上。

2 按兩下應用程式執行檔 (.air 檔) 來安裝應用程式。

預覽 Adobe AIR 應用程式

您隨時可以預覽構成 AIR 應用程式的任何頁面。也就是說，您不需要封裝應用程式，便可以看到安裝後的外觀。

1 確定您的 hello_world.html 頁面已經在 「Dreamweaver 文件」視窗中開啟。

2 按一下 「文件」工具列中的 「在瀏覽器中預覽 / 除錯」按鈕，接著選取 「在 AIR 中預覽」。

您也可以按 Ctrl+Shift+F12 (Windows) 或 Cmd+Shift+F12 (Macintosh)。

當您預覽這個頁面時，基本上您看到的是使用者將程式安裝在桌面上後所看到的開始頁面外觀。

使用 AIR SDK 建立您的第一個 HTML 類型 AIR 應用程式

若要親身體驗 Adobe® AIR® 的運作方式，請使用這些指示來建立和封裝簡單的 HTML 類型的 AIR "Hello World" 應用程式。

您必須安裝執行階段並設定 AIR SDK，才能夠開始。在本教學課程中，您要使用 AIR Debug Launcher (ADL) 和 AIR Developer Tool (ADT) 。ADL 和 ADT 是命令列公用程式，位於 AIR SDK 的 bin 目錄中 (請參閱第 15 頁 「安裝 AIR SDK」)。本教學課程假設您已經熟悉從命令列執行程式，並知道如何設定作業系統所需的 path 環境變數。

備註：如果您是 Adobe® Dreamweaver® 的使用者，請參閱第 27 頁「使用 Dreamweaver 建立您的第一個 HTML 類型 AIR 應用程式」。

備註：您只能針對 desktop 及 extendedDesktop 描述檔開發 HTML 類型的 AIR 應用程式。不支援行動描述檔。

建立專案檔案

每一個 HTML 類型 AIR 專案必須含有下列兩個檔案：指定應用程式中繼資料的應用程式描述器檔案，以及最上層 HTML 頁面。除了這些必要檔案外，這個專案還包含一個 JavaScript 程式碼檔案 AIRAliases.js，用來定義 AIR API 類別的易用別名變數。

1 建立名為 HelloWorld 的目錄來存放專案檔案。

2 建立名為 HelloWorld-app.xml 的 XML 檔案。

3 建立名為 HelloWorld.html 的 XML 檔案。

4 從 AIR SDK 的架構資料夾複製 AIRAliases.js 至專案目錄。

建立 AIR 應用程式描述器檔案

若要開始建立 AIR 應用程式，請建立具有下列結構的 XML 應用程式描述器檔案：

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

1 開啟 HelloWorld-app.xml 進行編輯。

2 加入根 <application> 元素，同時包括 AIR 命名空間特質：

<application xmlns="http://ns.adobe.com/air/application/2.7">：命名空間的最後一個區段 (「2.7」) 表示應用程式所需的執行階段版本。

3 加入 <id> 元素：

<id>examples.html.HelloWorld</id>：應用程式 ID 連同發行者 ID (由 AIR 衍生自用來簽署應用程式套件的憑證) 可識別應用程式的唯一性。應用程式 ID 的用途為安裝、存取私有應用程式檔案系統儲存目錄、存取私有加密儲存，以及應用程式之間的通訊。

4 新增 <versionNumber> 元素：

<versionNumber>0.1</versionNumber>：協助使用者判斷所安裝的應用程式版本。

備註：若是使用 AIR 2 或更早版本，您必須使用 <version> 元素而非 <versionNumber> 元素。

5 加入 <filename> 元素：

<filename>HelloWorld</filename>：用於應用程式可執行檔、安裝目錄和作業系統中對應用程式的其他參考名稱。

6 加入 <initialWindow> 元素，其中包含下列子元素，以指定初始應用程式視窗的屬性：

<content>HelloWorld.html</content>：識別供 AIR 載入的根 HTML 檔案。

<visible>true</visible>：可立即顯示視窗。

<width>400</width>：設定視窗寬度 (以像素為單位)。

<height>200</height>：設定視窗高度。

7 儲存檔案。完成的應用程式描述器檔案應該看起來與下列相似：

```
<?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>
```

這個範例僅設定一些可能的應用程式屬性。如需用來指定視窗顏色、視窗大小、透明度、預設安裝目錄、關聯檔案類型以及應用程式圖示等的全套應用程式屬性說明，請參閱 第 173 頁 「AIR 應用程式描述器檔案」。

建立應用程式 HTML 頁面

您現在必須建立一個簡單的 HTML 頁面，來當做 AIR 應用程式的主要檔案。

1 開啟 HelloWorld.html 以進行編輯。請加入下列 HTML 程式碼：

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

2 在 HTML 的 <head> 區段，匯入 AIRAliases.js 檔案：

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

AIR 會在 HTML 視窗物件定義一個名為 runtime 的屬性。runtime 屬性會使用類別的完整套件名稱來存取內建 AIR 類別。例如，在建立 AIR 檔案物件時，您可以在 JavaScript 中加入下列陳述式：

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

AIRAliases.js 檔案定義常用 AIR API 的易用別名。使用 AIRAliases.js 後，您可以縮短 File 類別的參考，如以下所示：

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

3 在 AIRAliases script 標籤下，新增另一個包含 JavaScript 函數的 script 標籤以處理 onLoad 事件：

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

appLoad() 函數僅呼叫 air.trace() 函數。當您使用 ADL 來執行應用程式時，追蹤訊息會列印到命令主控台。追蹤陳述式對於除錯非常實用。

4 儲存檔案。

HelloWorld.html 檔案現在看起來應該如下所示：

```
<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>
```

測試應用程式

若要從命令列執行和測試應用程式，請使用 AIR Debug Launcher (ADL) 公用程式。ADL 執行檔案位於 AIR SDK. 的 bin 目錄中。如果您尚未安裝 AIR SDK，請參閱第 15 頁「安裝 AIR SDK」。

1 開啟命令主控台或殼層。變更至您對這個專案建立的目錄。

2 執行下列命令：

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

隨即開啟 AIR 視窗，其中顯示您的應用程式。此外，主控台視窗會顯示 air.trace() 呼叫所產生的訊息。

如需詳細資訊，請參閱 第 173 頁 「AIR 應用程式描述器檔案」。

建立 AIR 安裝檔案

應用程式成功執行後，可以使用 ADT 公用程式將應用程式封裝成 AIR 安裝檔案。AIR 安裝檔案是包含所有應用程式檔案的存檔檔案，供您散佈給使用者。安裝封裝的 AIR 檔之前必須先安裝 Adobe AIR。

為了確保應用程式安全，所有 AIR 安裝檔案都必須經過數位簽名。您可以使用 ADT 或其他憑證產生工具來產生基本的自我簽署憑證，供開發使用。您也可以向商業憑證授權單位如 VeriSign 或 Thawte 購買商業程式碼簽署憑證。使用者安裝自我簽署 AIR 檔時，安裝過程中會顯示發行者 「不明」。這是因為自我簽署憑證僅保證 AIR 檔在建立後未曾變更。沒有辦法防止別人在偽裝的 AIR 檔中加入自我簽署來假冒是您的應用程式。強烈建議對公開發行的 AIR 檔案使用可驗證的商業憑證。如需 AIR 安全性問題的概觀，請參閱 AIR 安全性 (適用於 ActionScript 開發人員) 或 AIR security (適用於 HTML 開發人員)。

產生自我簽署憑證和金鑰組

❖ 從命令提示字元輸入下列命令 (ADT 執行檔案位於 AIR SDK 的 bin 目錄中)：

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

ADT 會產生名為 sampleCert.pfx 的金鑰儲存檔案，其中包含憑證以及相關的私密金鑰。

這個範例使用最少數目的特質設定憑證。金鑰類型必須是 1024-RSA 或 2048-RSA (請參閱第 161 頁 「簽署 AIR 應用程式」)。

建立 **AIR** 安裝檔案

❖ 從命令提示字元輸入下列命令 (在同一行中)：

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

將會出現提示，要求您提供金鑰儲存檔案密碼。

HelloWorld.air 引數是 ADT 產生的 AIR 檔。HelloWorld-app.xml 是應用程式描述器檔案。後續引數是您的應用程式使用的各個檔案。這個範例僅使用兩個檔案，但您可以加入任何數目的檔案和目錄。ADT 會驗證主要內容檔案 HelloWorld.html 是否包括在套件中，但是如果您忘記加入 AIRAliases.js，則您的應用程式將無法運作。

建立好 AIR 套件後，您可以按兩下套件檔案來安裝並執行應用程式。您還可以在殼層或命令視窗中輸入 AIR 檔案名稱當做命令。

後續步驟

HTML 和 JavaScript 程式碼在 AIR 中的行為，和在一般網頁瀏覽器中相同 (事實上，AIR 使用和 Safari 網頁瀏覽器相同的 WebKit 顯示引擎)。不過，您在 AIR 開發 HTML 應用程式時必須瞭解一些重要的差異。如需這些差異的詳細資訊，以及其他重要主題，請參閱 Programming HTML and JavaScript。

使用 Flex SDK 建立您的第一個桌上型 AIR 應用程式

若要親身體驗 Adobe® AIR® 的運作方式，請使用這些指示來利用 Flex SDK 建立簡單的 SWF 類型的 AIR "Hello World" 應用程式。這個教學課程會顯示如何使用 Flex SDK (Flex SDK 包括 AIR SDK) 提供的命令列工具來編譯、測試和封裝 AIR 應用程式。

您必須安裝執行階段並設定 Adobe® Flex™，才能夠開始。本教學課程使用 AMXMLC 編譯器、AIR Debug Launcher (ADL) 以及 AIR Developer Tool (ADT)。您可以在 Flex SDK 的 bin 目錄中找到這些程式 (請參閱第 16 頁「設定 Flex SDK」)。

建立 AIR 應用程式描述器檔案

本節描述如何建立應用程式描述器，這是內含以下結構的 XML 檔案：

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

1 建立名稱為 HelloWorld-app.xml 的 XML 檔案，然後儲存至專案目錄。

2 加入 <application> 元素，同時包括 AIR 命名空間特質：

<application xmlns="http://ns.adobe.com/air/application/2.7">：命名空間的最後一個區段 (「2.7」) 表示應用程式所需的執行階段版本。

3 新增 <id> 元素：

<id>samples.flex.HelloWorld</id> ：應用程式 ID 連同發行者 ID (由 AIR 衍生自用來簽署應用程式套件的憑證) 可識別應用程式的唯一性。建議的格式為逗點分隔的反向 DNS 樣式字串，例如 "com.company.AppName"。應用程式 ID 的用途為安裝、存取私有應用程式檔案系統儲存目錄、存取私有加密儲存，以及應用程式之間的通訊。

4 新增 <versionNumber> 元素：

<versionNumber>1.0</versionNumber>：協助使用者判斷所安裝的應用程式版本。

備註：若是使用 AIR 2 或更早版本，您必須使用 <version> 元素而非 <versionNumber> 元素。

5 加入 <filename> 元素：

<filename>HelloWorld</filename>：應用程式執行檔、安裝目錄以及作業系統中類似參考使用的名稱。

6 新增 <initialWindow> 元素，其中包含下列子元素，以指定初始應用程式視窗的屬性：

<content>HelloWorld.swf</content>：識別供 AIR 載入的根 SWF 檔案。

<visible>true</visible>：可立即顯示視窗。

<width>400</width>：設定視窗寬度 (以像素為單位)。

<height>200</height>：設定視窗高度。

7 儲存檔案。您的完整應用程式描述器檔案可能看起來如下：

```
<?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>
```

這個範例僅設定一些可能的應用程式屬性。如需用來指定視窗顏色、視窗大小、透明度、預設安裝目錄、關聯檔案類型以及應用程式圖示等的全套應用程式屬性說明，請參閱 第 173 頁 「AIR 應用程式描述器檔案」

撰寫應用程式程式碼

備註：SWF 類型的 AIR 應用程式可以使用 MXML 或 Adobe® ActionScript® 3.0 定義的主要類別。此範例使用 MXML 檔案來定義它的主要類別。使用主要 ActionScript 類別來建立 AIR 應用程式的程序都類似。不用將 MXML 檔案編譯成 SWF 檔案，您只要編譯 ActionScript 類別檔案即可。使用 ActionScript 的時候，主要類別必須擴充 flash.display.Sprite。

和所有 Flex 類型的應用程式一樣，使用 Flex 架構建立的 AIR 應用程式包含一個主要的 MXML 檔案。桌上型 AIR 應用程式會使用 WindowedApplication 組件當做根元素，而不是使用 Application 組件。WindowedApplication 組件提供各種屬性、方法和事件，用來控制應用程式及其初始視窗。在 AIR 不支援多個視窗的平台與描述檔上，請繼續使用 Application 組件。在行動 Flex 應用程式中，您也可以使用 View 或 TabbedViewNavigatorApplication 組件。

以下程序會建立 Hello World 應用程式：

1 使用文字編輯器，建立名稱為 HelloWorld.mxml 的檔案，然後加入以下的 MXML 程式碼：

```
<?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 接下來，將 Label 組件加入應用程式 (放到 WindowedApplication 標籤內)。

3 將 Label 組件的 text 屬性設定成 "Hello AIR"。

4 將佈局限制設成永遠保持在中央。

以下範例顯示到目前為止的程式碼：

```
<?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>
```

編譯應用程式

使用 amxmlc 編譯器將 MXML 程式碼編譯成 SWF 檔案，才能執行應用程式並進行除錯。您可以在 Flex SDK 資料夾的 bin 目錄中找到 amxmlc 編譯器。如果需要，您可以將電腦的路徑環境設定成包括 Flex SDK bin 目錄。設定這個路徑可以更方便地在命令列執行公用程式。

1 開啟命令殼層或終端機，然後瀏覽至 AIR 應用程式的專案資料夾。

2 輸入下列命令：

```
amxmlc HelloWorld.mxml
```

執行 amxmlc 會產生 HelloWorld.swf，它包含應用程式的已編譯程式碼。

備註：若應用程式未編譯，請修正語法或拼字錯誤。錯誤和警告會顯示在用來執行 AMXMLC 編譯器的主控台視窗。

如需詳細資訊，請參閱第 133 頁 「編譯 AIR 的 MXML 和 ActionScript 原始檔案」。

測試應用程式

若要從命令列執行並測試應用程式，請使用 AIR Debug Launcher (ADL)，並利用其應用程式描述器檔案來啟動應用程式 (您可以在 Flex SDK 資料夾的 bin 目錄中找到 ADL)。

❖ 從命令提示字元輸入以下命令：

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

產生的 AIR 應用程式會與下圖類似：

使用 Label 控制項的 horizontalCenter 和 verticalCenter 屬性，將文字放置在視窗的中央。移動或調整視窗大小的方式，與其他任何桌面應用程式相同。

如需詳細資訊，請參閱 第 137 頁 「AIR Debug Launcher (ADL)」。

建立 AIR 安裝檔案

應用程式成功執行後，可以使用 ADT 公用程式將應用程式封裝成 AIR 安裝檔案。AIR 安裝檔案是包含所有應用程式檔案的存檔檔案，供您散佈給使用者。安裝封裝的 AIR 檔之前必須先安裝 Adobe AIR。

為了確保應用程式安全，所有 AIR 安裝檔案都必須經過數位簽名。您可以使用 ADT 或其他憑證產生工具來產生基本的自我簽署憑證，供開發使用。您也可以向商業憑證授權單位購買商業程式碼簽署憑證。使用者安裝自我簽署 AIR 檔時，安裝過程中會顯示發行者 「不明」。這是因為自我簽署憑證僅保證 AIR 檔在建立後未曾變更。沒有辦法防止別人在偽裝的 AIR 檔中加入自我簽署來假冒是您的應用程式。強烈建議對公開發行的 AIR 檔案使用可驗證的商業憑證。如需 AIR 安全性問題的概觀，請參閱 AIR 安全性 (適用於 ActionScript 開發人員) 或 AIR security (適用於 HTML 開發人員)。

產生自我簽署憑證和金鑰組

❖ 從命令提示字元輸入以下命令 (您可以在 Flex SDK 的 bin 目錄找到 ADT 執行檔)：

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

這個範例使用最少數目的特質設定憑證。金鑰類型必須是 1024-RSA 或 2048-RSA (請參閱第 161 頁 「簽署 AIR 應用程式」)。

建立 AIR 套件

❖ 從命令提示字元輸入下列命令 (在同一行中)：

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

將會出現提示，要求您提供金鑰儲存檔案密碼。輸入密碼並按 Enter 鍵。基於安全理由，不會顯示密碼字元。

HelloWorld.air 引數是 ADT 產生的 AIR 檔。HelloWorld-app.xml 是應用程式描述器檔案。後續引數是您的應用程式使用的各個檔案。這個範例僅使用三個檔案，但您可以加入任何數目的檔案和目錄。

建立好 AIR 套件後，您可以按兩下套件檔案來安裝並執行應用程式。您還可以在殼層或命令視窗中輸入 AIR 檔案名稱當做命令。

如需詳細資訊，請參閱第 47 頁 「封裝桌上型 AIR 安裝檔案」。

使用 Flex SDK 建立您的第一個 AIR for Android 應用程式

首先，您必須已經安裝和設定 AIR 與 Flex SDK。這個教學課程使用 Flex SDK 中的 AMXMLC 編譯器、AIR Debug Launcher (ADL) 以及 AIR SDK 中的 AIR Developer Tool (ADT)。請參閱第 16 頁 「設定 Flex SDK」。

您必須另從 Android 網站下載和安裝 Android SDK，如需詳細資訊，請參閱：Android 開發人員：安裝 SDK。

備註：如需 iPhone 開發的資訊，請參閱使用 Flash Professional CS5 建立 Hello World iPhone 應用程式。

建立 AIR 應用程式描述器檔案

本節描述如何建立應用程式描述器，這是內含以下結構的 XML 檔案：

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

1 建立名稱為 HelloWorld-app.xml 的 XML 檔案，然後儲存至專案目錄。

2 加入 <application> 元素，同時包括 AIR 命名空間特質：

<application xmlns="http://ns.adobe.com/air/application/2.7">：命名空間的最後一個區段 (「2.7」) 表示應用程式所需的執行階段版本。

3 新增 <id> 元素：

<id>samples.android.HelloWorld</id> ：應用程式 ID 連同發行者 ID (由 AIR 衍生自用來簽署應用程式套件的憑證) 可識別應用程式的唯一性。建議的格式為句點分隔的反向 DNS 樣式字串，例如 "com.company.AppName"。

4 加入 <versionNumber> 元素：

<versionNumber>0.0.1</versionNumber>：協助使用者判斷所安裝的應用程式版本。

5 新增 <filename> 元素：

<filename>HelloWorld</filename>：應用程式執行檔、安裝目錄以及作業系統中類似參考使用的名稱。

6 新增 <initialWindow> 元素，其中包含下列子元素，以指定初始應用程式視窗的屬性：

<content>HelloWorld.swf</content>：識別供 AIR 載入的根 HTML 檔案。

7 新增 <supportedProfiles> 元素。

<supportedProfiles>mobileDevice</supportedProfiles> 指定應用程式只能在行動描述檔中執行。

8 儲存檔案。您的完整應用程式描述器檔案可能看起來如下：

```
<?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>
```

這個範例僅設定一些可能的應用程式屬性。應用程式描述器檔案中還有其他您可以使用的設定。例如，您可以將 <fullScreen>true</fullScreen> 新增至 initialWindow 元素以建立全螢幕應用程式。若要在 Android 上啟用遠端除錯和存取控制的功能，您也必須將 Android 權限加入應用程式描述器。這個簡單的應用程式不需設定權限，因此您現在不需要新增權限。

如需詳細資訊，請參閱第 61 頁 「設定行動應用程式屬性」。

撰寫應用程式程式碼

建立名為 HelloWorld.as 的檔案，並使用文字編輯器新增下列程式碼：

```
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 );
        }
    }
}
```

編譯應用程式

使用 amxmlc 編譯器將 MXML 程式碼編譯成 SWF 檔案，才能執行應用程式並進行除錯。您可以在 Flex SDK 資料夾的 bin 目錄中找到 amxmlc 編譯器。如果需要，您可以將電腦的路徑環境設定成包括 Flex SDK bin 目錄。設定這個路徑可以更方便地在命令列執行公用程式。

1 開啟命令殼層或終端機，然後瀏覽至 AIR 應用程式的專案資料夾。

2 輸入下列命令：

```
amxmlc HelloWorld.as
```

執行 amxmlc 會產生 HelloWorld.swf，它包含應用程式的已編譯程式碼。

備註：若應用程式未編譯，請修正語法或拼字錯誤。錯誤和警告會顯示在用來執行 AMXMLC 編譯器的主控台視窗。

如需詳細資訊，請參閱第 133 頁 「編譯 AIR 的 MXML 和 ActionScript 原始檔案」。

測試應用程式

若要從命令列執行並測試應用程式，請使用 AIR Debug Launcher (ADL)，並利用其應用程式描述器檔案來啟動應用程式 (您可以在 AIR 與 Flex SDK 的 bin 目錄中找到 ADL)。

❖ 從命令提示字元輸入以下命令：

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

如需詳細資訊，請參閱第 86 頁 「使用 ADL 進行裝置模擬」。

建立 APK 套件檔案

應用程式成功執行後，可以使用 ADT 公用程式將應用程式封裝成 APK 套件檔案。APK 套件檔案是原生 Android 應用程式檔案格式，可直接散佈給使用者。

所有 Android 應用程式都必須經過簽署。不同於 AIR 檔案，Android 應用程式通常會使用自我簽署的憑證來簽署。Android 作業系統不會嘗試建立應用程式開發人員的身分。您可以使用 ADT 產生的憑證來簽署 Android 套件。送出至 Android Market 的應用程式憑證有效期必須至少為 25 年。

產生自我簽署憑證和金鑰組

❖ 從命令提示字元輸入以下命令 (您可以在 Flex SDK 的 bin 目錄找到 ADT 執行檔)：

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

這個範例使用最少數目的特質設定憑證。金鑰類型必須是 1024-RSA 或 2048-RSA (請參閱 第 149 頁 「ADT certificate 命令」)。

建立 AIR 套件

❖ 從命令提示字元輸入下列命令 (在同一行中)：

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

將會出現提示，要求您提供金鑰儲存檔案密碼。輸入密碼並按 Enter 鍵。

如需詳細資訊，請參閱第 80 頁 「封裝行動 AIR 應用程式」。

安裝 AIR 執行階段

您可以在裝置上安裝從 Android Market 取得的 AIR 執行階段最新版本。您也可以在裝置或 Android 模擬器上安裝 SDK 中包括的執行階段。

❖ 從命令提示字元輸入下列命令 (在同一行中)：

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

將 -platformsdk 旗標設定為您的 Android SDK 目錄 (指定 tools 資料夾的上層資料夾)。

ADT 會安裝 SDK 中的 Runtime.apk。

如需詳細資訊，請參閱第 93 頁 「安裝用於開發的 AIR 執行階段與應用程式」。

安裝 AIR 應用程式

❖ 從命令提示字元輸入下列命令 (在同一行中)：

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

將 -platformsdk 旗標設定為您的 Android SDK 目錄 (指定 tools 資料夾的上層資料夾)。

如需詳細資訊，請參閱第 93 頁 「安裝用於開發的 AIR 執行階段與應用程式」。

您可以在裝置或模擬器的螢幕上，輕點應用程式圖示以啟動應用程式。

第 6 章 開發桌上型 AIR 應用程式

開發桌上型 AIR 應用程式的工作流程

開發 AIR 應用程式的基本工作流程與多數傳統開發模型相同：撰寫程式碼、編譯、測試，以及在週期結束時封裝為安裝程式檔案。

您可以使用 Flash、Flex 與 ActionScript 撰寫應用程式程式碼，並使用 Flash Professional、Flash Builder 或 mxmlc 與 compc 命令列編譯器來進行編譯。您也可以使用 HTML 與 JavaScript 來撰寫應用程式程式碼，並略過編譯步驟。

您可以使用 ADL 工具來測試桌上型 AIR 應用程式，ADL 工具毋須封裝與安裝便能執行應用程式。Flash Professional、Flash Builder、Dreamweaver 與 Aptana IDE 皆整合 Flash 除錯程式。從命令列使用 ADL 時，也可手動啟動除錯程式工具 FDB。ADL 本身不會顯示錯誤與追蹤陳述式輸出。

所有 AIR 應用程式都必須封裝成為安裝檔案。建議使用跨平台 AIR 檔案格式，除非：

- 您必須存取平台式 API，例如 NativeProcess 類別。
- 您的應用程式會使用原生擴充功能。

在必須存取特定平台 API 的情況下，可將 AIR 應用程式封裝為特定平台的原生安裝程式檔案。

SWF 類型應用程式

1. 撰寫 MXML 或 ActionScript 程式碼。
2. 建立所需的資源，例如，圖示點陣圖檔案。
3. 建立應用程式描述器。
4. 編譯 ActionScript 程式碼。
5. 測試應用程式。
6. 使用 air 目標封裝並以 AIR 檔案登入。

HTML 類型應用程式

1. 撰寫 HTML 與 JavaScript 程式碼。
2. 建立所需的資源，例如，圖示點陣圖檔案。
3. 建立應用程式描述器。
4. 測試應用程式。
5. 使用 air 目標封裝並以 AIR 檔案登入。

建立 AIR 應用程式原生安裝程式

1. 撰寫程式碼 (ActionScript 或 HTML 與 JavaScript)。
2. 建立所需的資源，例如，圖示點陣圖檔案。
3. 建立應用程式描述器，指定 extendedDesktop 描述檔。
4. 編譯任何 ActionScript 程式碼。
5. 測試應用程式。

6 使用原生目標在每個目標平台上封裝應用程式。

備註：必須在目標平台上建立該平台的原生安裝程式。舉例，您無法在 Mac 上建立 Windows 安裝程式。您可使用虛擬機器，例如 VMW，在相同的電腦硬體上執行多個平台。

使用固定執行階段組合包來建立 AIR 應用程式

1 撰寫程式碼 (ActionScript 或 HTML 與 JavaScript)。

2 建立所需的資源，例如，圖示點陣圖檔案。

3 建立應用程式描述器，指定 extendedDesktop 描述檔。

4 編譯任何 ActionScript 程式碼。

5 測試應用程式。

6 使用組合包目標在每個目標平台上封裝應用程式。

7 使用組合包檔案建立安裝程式 (AIR SDK 不提供建立建立此類安裝程式的工具，但另有許多可用的協力廠商工具套件)。

備註：必須在目標平台上建立該平台的組合包。舉例，您無法在 Mac 上建立 Windows 組合包。您可使用虛擬機器，例如 VMW，在相同的電腦硬體上執行多個平台。

設定桌上型應用程式屬性

設定應用程式描述器檔案中的基本應用程式屬性。此部分涵蓋與桌上型 AIR 應用程式相關的屬性。第 173 頁 「AIR 應用程式描述器檔案」中完整說明應用程式描述器檔案的元素。

必要的 AIR 執行階段版本

使用應用程式描述器檔案的命名空間，指定應用程式所需的 AIR 執行階段版本。

application 元素中指派的命名空間，大致上決定了應用程式可使用的功能。例如，如果應用程式使用 AIR 1.5 命名空間，且使用者已安裝 AIR 3.0，則應用程式會出現 AIR 1.5 的行為 (即使在 AIR 3.0 中已變更該行為)。只有當您變更命名空間且發佈更新，應用程式才能存取新行為與功能。安全性與 WebKit 變更為此一原則的主要例外。

使用根 application 元素的 xmlns 特質，指定命名空間：

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

更多說明主題

第 178 頁 「application」

應用程式身分識別

針對發佈的每個應用程式，有數種設定必須是唯一的。這些唯一的設定包括 ID、名稱與檔案名稱。

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

更多說明主題

第 193 頁「id」

第 188 頁「filename」

第 200 頁「name」

應用程式版本

在 AIR 2.5 之前的 AIR 版本中，在 version 元素中指定應用程式。您可以使用任何字串。AIR 執行階段不會解譯字串；“2.0” 不會被視為高於 “1.0” 的版本。

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

在 AIR 2.5 與更新的版本中，在 versionNumber 元素中指定應用程式版本。version 元素已不再可供使用。指定 versionNumber 的值時，必須使用最多三個號碼的序列，號碼之間以點分隔，例如：“0.1.2”。版本號碼的每個區段最多可以有三位數字。(亦即，“999.999.999” 為允許的最大版本號碼)。號碼中毋須包括所有三個區段；“1” 與 “1.0” 都是合法的版本號碼。

您也可以使用 versionLabel 元素，指定版本的標籤。新增版本標籤時，在 AIR 應用程式安裝程式對話方塊等處，會顯示標籤以取代版本號碼。

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

更多說明主題

第 207 頁「version」

第 208 頁「versionLabel」

第 208 頁「versionNumber」

主要視窗屬性

當 AIR 在桌上型電腦上啟動應用程式時，會建立視窗並將主要 SWF 檔案或 HTML 頁面載入視窗。AIR 使用 initialWindow 元素中的子元素，控制此初始視窗的初始外觀與行為。

- **content** — initalWindow 元素中 content 子元素的主要應用程式 SWF 檔案。以桌上型描述檔中的裝置為目標時，可使用 SWF 或 HTML 檔案。

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

  您必須將檔案包括在 AIR 套件中 (使用 ADT 或您的 IDE)。只是參照應用程式描述器中的名稱，不會將檔案自動包括在套件中。

- **depthAndStencil** — 指定使用深度或模板緩衝區。您通常會在使用 3D 內容時使用這些緩衝區。

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

- **height** — 初始視窗的高度。
- **maximizable** — 是否顯示最大化視窗的系統顏色。
- **maxSize** — 允許的大小上限。
- **minimizable** — 是否顯示用於最小化視窗的系統顏色。
- **minSize** — 允許的大小下限。

- **renderMode** — 在 AIR 3 或更新的版本中，可將桌上型應用程式的演算模式設定為 auto、cpu、direct 或 gpu。在舊版 AIR，會忽略桌上型平台中的此一設定。您無法在執行階段變更 renderMode 設定。
  - auto — 基本上與 cpu 模式相同。
  - cpu — 會顯示並複製演算的物件，以顯示軟體中的記憶體。僅有在全螢幕的視窗模式時，才可使用 StageVideo。Stage3D 會使用軟體演算器。
  - direct — 顯示物件是由執行階段軟體所顯示，但複製演算影格以顯示硬體則會加速記憶體 (位圖複製)。可使用 StageVideo。如果可以的話，Stage3D 會使用硬體加速。如果將視窗透明度設為 true，則視窗會 「回復」至軟體顯示與位圖複製。

    備註：若要在行動平台上使用 AIR 來利用 Flash 內容的 GPU 加速，Adobe 建議您使用 renderMode="direct" (即 Stage3D)，而不要使用 renderMode="gpu"。Adobe 正式支援並建議使用下列 Stage3D 架構：Starling (2D) 和 Away3D (3D)。如需有關 Stage3D 和 Starling/Away3D 的詳細資訊，請參閱 http://gaming.adobe.com/getstarted/。
  - gpu — 使用硬體加速 (若有的話)。
- **requestedDisplayResolution** — 在具有高解析度螢幕的 MacBook Pro 電腦上，應用程式應該使用 「標準」還是 「高」解析度模式。在所有其他平台上，值會遭到忽略。如果值為 「標準」，則每一舞台像素會顯示為螢幕上的 4 像素。如果值為 「高」，則每一舞台像素相當於螢幕上的單一實際像素。指定的值可用於所有應用程式視窗。在 AIR 3.6 及更新版本中，requestedDisplayResolution 元素可以用於桌上型 AIR 應用程式 (做為 intialWindow 元素的子元素)。
- **resizable** — 是否顯示調整視窗大小的系統顏色。
- **systemChrome** — 是否使用標準的作業系統視窗版面配置。執行階段無法變更視窗的 systemChrome 設定。
- **title** — 視窗的標題。
- **transparent** — 視窗是否與背景進行 Alpha 混合。若開啟透明功能，視窗將無法使用系統顏色。執行階段無法變更視窗的 transparent 設定。
- **visible** — 建立視窗時，是否顯示視窗。根據預設，一開始不會顯示視窗，而是讓應用程式可以在繪製內容後再顯示視窗。
- **width** — 視窗的寬度。
- **x** — 視窗的水平位置。
- **y** — 視窗的垂直位置。

更多說明主題

第 183 頁 「content」
第 184 頁 「depthAndStencil」
第 192 頁 「height」
第 199 頁 「maximizable」
第 199 頁 「maxSize」
第 200 頁 「minimizable」
第 200 頁 「minimizable」
第 200 頁 「minSize」
第 202 頁 「renderMode」
第 203 頁 「requestedDisplayResolution」
第 204 頁 「resizable」
第 206 頁 「systemChrome」

第 207 頁「title」
第 207 頁「transparent」
第 208 頁「visible」
第 209 頁「width」
第 209 頁「x」
第 210 頁「y」

桌上型功能

下列元素控制桌上型安裝與更新功能。

- customUpdateUI — 讓您自行提供更新應用程式的對話方塊。若設為 false (預設)，則使用標準的 AIR 對話方塊。
- fileTypes — 指定應用程式要註冊作為預設開啟應用程式的檔案類型。如果其他應用程式已經是某個檔案類型的預設開啟應用程式，則 AIR 不會覆寫現有的註冊。不過，應用程式可以在執行階段使用 NativeApplication 物件的 setAsDefaultApplication() 方法來覆寫註冊。建議在覆寫現有檔案類型關聯之前，取得使用者的同意。

  備註：當您 (使用 -bundle 目標) 將應用程式封裝為固定執行階段組合包時，會忽略檔案類型登錄。若要登錄指定的檔案類型，您必須建立執行登錄的安裝程式。
- installFolder — 指定相對於標準應用程式安裝資料夾的路徑，以安裝應用程式。您可以使用此設定，提供自訂資料夾名稱，以及將多個應用程式群組在同一個資料夾中。
- programMenuFolder — 指定 Windows「所有程式」選單的選單階層。您可以使用此設定，將多個應用程式群組在同一個選單中。若未指定選單資料夾，應用程式捷徑將直接新增至主要選單。

更多說明主題

第 184 頁「customUpdateUI」
第 189 頁「fileTypes」
第 196 頁「installFolder」
第 201 頁「programMenuFolder」

支援的描述檔

如果應用程式僅適用於桌上型，則可以從支援的描述檔清單中排除特定描述檔，以避免讓應用程式安裝在其他描述檔中的裝置上。如果應用程式使用 NativeProcess 類別或原生擴充功能，則必須支援 extendedDesktop 描述檔。

如果應用程式描述器不包含 supportedProfile 元素，則預設應用程式支援所有定義的描述檔。若要限制應用程式侷限於特定的描述檔清單，請列出以空白區隔的描述檔：

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

如需 desktop 與 extendedDesktop 描述檔中支援的 ActionScript 類別的清單，請參閱第 212 頁「不同描述檔的功能」。

更多說明主題

第 205 頁「supportedProfiles」

必須要有原生擴充功能

支援 extendedDesktop 描述檔的應用程式可以使用原生擴充功能。

請在應用程式描述器中宣告 AIR 應用程式會使用的所有原生擴充功能。下列範例說明指定兩個必要原生擴充功能的語法：

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

extensionID 元素的值與擴充功能描述器檔案中的 id 元素相同。擴充功能描述器檔案是一個名為 extension.xml 的 XML 檔。它是封裝於您從原生擴充功能開發人員得到的 ANE 檔案。

應用程式圖示

在桌上型電腦上，應用程式描述器中指定的圖示，將用於應用程式檔案、捷徑與程式選單圖示。提供的應用程式圖示應為一組 16x16、32x32、48x48 與 128x128 像素 PNG 影像。在應用程式描述器檔案的圖示元素中，指定圖示檔案的路徑：

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

如果未提供特定大小的圖示，將使用下一個最大大小的圖示，並縮放為適合的大小。如果未提供任何圖示，將使用預設系統圖示。

更多說明主題

第 192 頁「icon」

第 193 頁「imageNxN」

忽略的設定

桌上型電腦上的應用程式將忽略適用於行動描述檔功能的應用程式設定。忽略的設定如下：

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

除錯桌上型 AIR 應用程式

如果使用 Flash Builder、Flash Professional 或 Dreamweaver 等 IDE 開發應用程式，通常有內建的除錯工具。您可以在除錯模式中啟動應用程式，便能為應用程式除錯。如果不是使用直接支援除錯的 IDE，則可使用 AIR Debug Launcher (ADL) 與 Flash Debugger (FDB) 來協助除錯應用程式。

更多說明主題

De Monsters：Monster Debugger

第 243 頁「使用 AIR HTML Introspector 除錯」

使用 ADL 執行應用程式

您可以使用 ADL，毋須封裝與安裝 AIR 應用程式，便能執行 AIR 應用程式。如下列範例所示，將應用程式描述器檔案作為參數傳送至 ADL (應用程式的 ActionScript 程式碼必須先編譯)：

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

ADL 列印追蹤陳述式、執行階段例外與 HTML 剖析錯誤至終端機視窗。如果 FDB 處理程序正在等待內送連線，ADL 將連線至除錯程式。

您也可使用 ADL 來為使用原生擴充功能的 AIR 應用程式除錯。例如：

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

更多說明主題

第 137 頁「AIR Debug Launcher (ADL)」

列印追蹤陳述式

若要將追蹤陳述式列印至用於執行 ADL 的主控台，請使用 trace() 函數將追蹤陳述式加入程式碼中。

備註：如果您的 trace() 陳述式未顯示在主控台上，請確定您尚未在 mm.cfg 檔案中指定 ErrorReportingEnable 或 TraceOutputFileEnable。如需有關這個檔案之平台特定位置的詳細資訊，請參閱編輯 mm.cfg 檔案。

ActionScript 範例：

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

JavaScript 範例：

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

在 JavaScript 程式碼，您可以使用 alert() 和 confirm() 函數顯示應用程式的除錯訊息。此外，語法有錯誤的行號和未捕捉的 JavaScript 例外也都會列印至主控台。

備註：若要使用 JavaScript 範例顯示的 air 前置詞，您必須將 AIRAliases.js 檔案匯入頁面中。這個檔案位於 AIR SDK 架構目錄內。

連接至 Flash 除錯程式 (FDB)

若要使用 Flash 除錯程式來除錯 AIR 應用程式，請啟動 FDB 工作階段，然後使用 ADL 來啟動應用程式。

備註：在 SWF 類型的 AIR 應用程式中，ActionScript 原始檔案必須使用 -debug 旗標進行編譯。(在 Flash Professional，請勾選「發佈設定」對話方塊中的「允許除錯」選項)。

1. 啟動 FDB。您可以在 Flex SDK 資料夾的 bin 目錄中找到 FDB 程式。

   主控台會顯示 FDB 提示：<fdb>

2. 執行 run 命令：<fdb>run [Enter]
3. 在不同的命令或殼層主控台中，啟動應用程式的除錯版本：

   ```
   adl myApp.xml
   ```

4 使用 FDB 命令設定所需的中斷點。

5 輸入：continue [Enter]

如果 AIR 應用程式屬於 SWF 類型，除錯程式就只會控制 ActionScript 程式碼的執行，而如果屬於 HTML 類型，除錯程式就只會控制 JavaScript 程式碼的執行。

若要在不連接除錯程式的情況下執行 ADL，請納入 -nodebug 選項：

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

如需有關 FDB 命令的基本資訊，請執行 help 命令：

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

如需有關 FDB 命令的詳細資訊，請參閱 Flex 文件中的使用除錯程式的命令列命令。

封裝桌上型 AIR 安裝檔案

每一個 AIR 應用程式至少都要有一個應用程式描述器檔案和主要 SWF 或 HTML 檔。隨應用程式安裝的其他任何資源也必須封裝在 AIR 檔案中。

此文章討論使用 SDK 包含的命令列工具來封裝 AIR 應用程式。如需使用 Adobe 編寫工具之一來封裝應用程式的詳細資訊，請參閱下列項目：

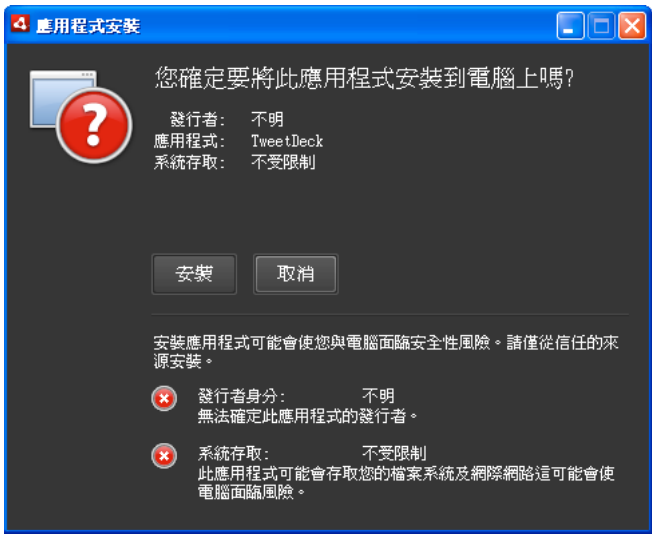
- Adobe® Flex® Builder™，請參閱 Packaging AIR applications with Flex Builder。
- Adobe® Flash® Builder™，請參閱 Packaging AIR applications with Flash Builder。
- Adobe® Flash® Professional，請參閱 Adobe AIR 的發佈功能。
- Adobe® Dreamweaver®，請參閱使用 Dreamweaver 建立 AIR 應用程式。

所有 AIR 安裝程式檔案都必須使用數位憑證來加以簽署。AIR 安裝程式會使用這項簽名，確認應用程式檔案在您簽署之後並未遭到修改。您可以使用憑證授權單位提供的程式碼簽署憑證，或使用自我簽署的憑證。

使用由受信任憑證授權單位發行的憑證時，應用程式的使用者便可以確認您的發行者身分。安裝對話方塊會反映出您的身分已由憑證授權單位驗證。

已由受信任憑證簽署的應用程式安裝確認對話方塊

當您使用自我簽署的憑證時，使用者便無法驗證您的簽署者身分。自我簽署憑證也無法完全保證套件未遭到竄改 (這是因為在合法安裝檔案送達使用者之前，該安裝檔案可能已被偽造的檔案所取代)。而安裝對話方塊會反映出無法驗證發行者的身分。當使用者在安裝您的應用程式時，便會承擔非常高的安全性風險：



已由自我簽署憑證簽署的應用程式安裝確認對話方塊

您可以使用 ADT 的 -package 命令，以單一步驟完成 AIR 檔的封裝及簽署。您也可以使用 -prepare 命令建立未經簽署的中繼套件，然後再另外使用 -sign 命令簽署此中繼套件。

備註：Java 1.5 版及以上版本在用來保護 PKCS12 憑證檔案的密碼中，不接受高位元 ASCII 字元。當您建立或匯出程式碼簽署憑證檔案時，請只在密碼中使用一般的 ASCII 字元。

簽署安裝套件時，ADT 會自動聯繫時間戳記授權伺服器以驗證時間。此時間戳記資訊會包含在 AIR 檔中。之後，您隨時可以安裝包含經過驗證之時間戳記的 AIR 檔。如果 ADT 無法連接至時間戳記伺服器，封裝作業就會取消。您可以覆寫時間戳記選項，但是如果沒有時間戳記，在用於簽署安裝檔案的憑證過期之後，就無法再安裝 AIR 應用程式。

如果您要建立套件來更新現有的 AIR 應用程式，就必須利用和原始應用程式相同的憑證來簽署該套件。如果原始憑證已被更新或在過去 180 天內過期，或如果您想要變更為新的憑證，則您可以套用移轉簽名。移轉簽名會同時使用新的與舊的憑證來簽署應用程式 AIR 檔案。使用 -migrate 命令，如 第 148 頁 「ADT migrate 命令」所述套用移轉簽名。

重要事項：在原始憑證過期之後，您可以在 180 天寬限期內套用移轉簽名。若不使用移轉簽名，現有使用者就必須先解除安裝現有的應用程式，然後才能安裝您的新版本。寬限期只適用於在應用程式描述器命名空間中，指定 AIR 1.5.3 版或更新版本的應用程式。若指定的是較早版本的 AIR 執行階段，則無寬限期。

AIR 1.1 之前的版本並不支援移轉簽名。您必須利用 SDK 1.1 版或更新版本封裝應用程式，才能套用移轉簽名。

使用 AIR 檔案部署的應用程式也稱為桌上型描述檔應用程式。如果應用程式描述器檔案不支援桌上型描述檔，您將無法使用 ADT 來封裝 AIR 應用程式的原生安裝程式。您可以在應用程式描述器檔案中使用 supportedProfiles 元素來限制此描述檔。請參閱第 211 頁 「裝置描述檔」與 第 205 頁 「supportedProfiles」。

備註：應用程式描述器檔案中的設定會判斷 AIR 應用程式的識別及預設安裝路徑。請參閱 第 173 頁 「AIR 應用程式描述器檔案」。

發行者 ID

AIR 1.5.3 已不再使用發行者 ID。新的應用程式 (利用 AIR 1.5.3 或更新版本所發行) 不需要亦不應該指定發行者 ID。

當您更新以較早版本 AIR 發行的應用程式時，必須在應用程式描述器檔案中指定原始發行者 ID。否則，應用程式的安裝版本與更新版本會被視為是不同的應用程式。如果您使用不同的 ID 或省略 publisherID 標籤，那麼使用者就必須先解除安裝之前的版本，然後才能安裝新版的應用程式。

若要判斷原始發行者 ID，請到原始應用程式安裝處的 META-INF/AIR 子目錄中找出 publisherid 檔案。這個檔案中的字串就是發行者 ID。應用程式描述器必須在應用程式描述器的命名空間宣告中指定 AIR 1.5.3 執行階段 (或更新版本)，以便手動指定發行者 ID。

對於在 AIR 1.5.3 之前所發佈的應用程式 (或在應用程式描述器命名空間中指定較早版本的 AIR，但實際上卻使用 AIR 1.5.3 SDK 來發佈)，其發行者 ID 是根據簽署憑證所計算出來的。此 ID 將與應用程式 ID 搭配，以產生應用程式的識別。發行者 ID (如果有的話) 可用於以下用途：

- 驗證 AIR 檔案是更新版本，而不是要安裝的新應用程式
- 加密本機儲存區加密金鑰的一部分
- 應用程式儲存目錄路徑的一部分
- 本機連線之連線字串的一部分
- 識別字串的一部分，AIR 內部瀏覽器 API 用來呼叫應用程式
- OSID 的一部分 (建立自訂安裝 / 解除安裝程式時使用)

在 AIR 1.5.3 之前，如果您使用新的或更新的憑證，利用移轉簽名來簽署應用程式更新，那麼應用程式的發行者 ID 便可能會變更。當發行者 ID 變更時，所有與此 ID 相關的 AIR 功能也會隨之變更行為。例如，現有加密本機儲存區中的資料將無法存取，而且任何建立該應用程式本機連線的 Flash 或 AIR 實體，都必須在連線字串中使用新的 ID。

在 AIR 1.5.3 或以上版本中，發行者 ID 並非以簽署憑證為基礎，只有當應用程式描述器中含有 publisherID 標籤時，才會指定發行者 ID。如果在更新 AIR 套件中指定的發行者 ID 不符合目前的發行者 ID，將無法更新應用程式。

使用 ADT 封裝

您可以使用 AIR ADT 命令列工具來封裝 AIR 應用程式。封裝之前，必須編譯所有的 ActionScript、MXML 與任何擴充程式碼。您也必須具有程式碼簽署憑證。

如需 ADT 命令與選項的詳細參考資訊，請參閱 第 142 頁 「AIR Developer Tool (ADT)」。

建立 AIR 套件

若要建立 AIR 套件，請使用 ADT package 命令，將發行組建的目標類型設為 air。

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

此範例假設 ADT 工具的路徑位於命令列殼層路徑定義中。(如需說明，請參閱 第 263 頁 「Path 環境變數」)。

您必須從包含應用程式檔案的目錄中執行命令。範例中的應用程式檔案為 myApp-app.xml (應用程式描述器檔案)、myApp.swf 與圖示目錄。

如圖所示執行命令時，ADT 將提示您輸入金鑰儲存密碼。(您輸入的密碼字元不會永遠顯示；完成輸入時請按 Enter)。

從 AIRI 檔案建立 AIR 套件

您可以建立簽署 AIRI 檔案，以建立可安裝的 AIR 套件：

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

封裝桌上型原生安裝程式

如 AIR 2 中，您可以使用 ADT 建立原生應用程式安裝程式，進而散佈 AIR 應用程式。例如，您可以建立 EXE 安裝程式以在 Windows 上發佈 AIR 應用程式。您可以建立 DMG 安裝程式以在 Mac OS 上發佈 AIR 應用程式。在 AIR 2.5 和 AIR 2.6 上，您可以建立 DEB 或 RPM 安裝程式檔案，以便在 Linux 上發佈 AIR 應用程式。

隨原生應用程式安裝程式安裝的應用程式稱為延伸桌上型描述檔應用程式。如果應用程式描述器檔案不支援桌上型延伸描述檔，您便無法使用 ADT 來封裝 AIR 應用程式的原生安裝程式。您可以在應用程式描述器檔案中使用 supportedProfiles 元素來限制此描述檔。請參閱第 211 頁 「裝置描述檔」與 第 205 頁 「supportedProfiles」。

有兩種方法可建立 AIR 應用程式的原生安裝程式版本：

- 您可以根據應用程式描述器檔案與其他來源檔案，建立原生安裝程式 (其他來源檔案可能包括 SWF 檔案、HTML 檔案以及其他資源)。
- 您可以根據 AIR 檔案或根據 AIRI 檔案，建立原生安裝程式。

您必須在想要產生原生安裝程式的作業系統上使用 ADT。這表示若要建立 Windows 適用的 EXE 檔，請在 Windows 上執行 ADT。若要建立 Mac OS 適用的 DMG 檔，請在 Mac OS 上執行 ADT。若要建立 Linux 適用的 DEB 或 RPG 檔案，請在 Linux 上從 AIR 2.6 SDK 執行 ADT。

當您建立原生安裝程式以發佈 AIR 應用程式時，該應用程式具有下列功能：

- 應用程式可使用 NativeProcess 類別來啟動原生處理程序並與之互動。如需詳細資訊，請參閱下列其中一項：
  - 與 AIR 中的原生處理程序通訊 (適用於 ActionScript 開發人員)
  - Communicating with native processes in AIR (適用於 HTML 開發人員)
- 它可使用原生擴充功能。
- 使用 File.openWithDefaultApplication() 方法可使用預設的系統應用程式來開啟檔案，而不論其檔案類型。(不是以原生安裝程式安裝的應用程式會有部分限制。如需詳細資訊，請參閱語言參考中的 File.openWithDefaultApplication() 項目)。

不過，封裝為原生安裝程式時，應用程式會喪失 AIR 檔案格式的部分優點。單一檔案將無法再散佈至所有桌上型電腦。內建更新功能 (以及更新程式架構) 無法運作。

使用者按兩下原生安裝程式檔案時，便會安裝 AIR 應用程式。如果電腦上尚未安裝所需的 Adobe AIR 版本，安裝程式便會先從網路下載該版本，然後予以安裝。如果沒有可取得正確 Adobe AIR 版本 (如有需要) 的網路連線，安裝就會失敗。此外，如果 Adobe AIR 2 不支援該作業系統，安裝也會失敗。

備註： 如果您希望檔案可以在安裝的應用程式中執行，請在封裝應用程式時，確認該檔案可在檔案系統上執行 (您可以在 Mac 和 Linux 上，視需要使用 chmod 設定可執行旗標)。

從應用程式來源檔案建立原生安裝程式

若要從應用程式的來源檔案建立原生安裝程式，請使用 -package 命令和下列語法 (單行命令列)：

```
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 ...] ...
```

此語法和封裝 AIR 檔案 (不包含原生安裝程式) 的語法相似，不過有一些差異：

- 將 -target native 選項加入命令 (如果指定 -target air，則 ADT 會產生 AIR 檔案，而不是原生安裝程式檔案)。
- 指定目標 DMG 或 EXE 檔案做為 installer_file。
- 或者是，在 Windows 中加入第二組簽署選項，以 [WINDOWS_INSTALLER_SIGNING_OPTIONS] 語法表示。在 Windows 中，除了簽署 AIR 檔案之外，您還可以簽署 Windows Installer 檔案。請使用和簽署 AIR 檔案時相同的憑證類型與簽署選項語法 (請參閱 第 153 頁 「ADT 程式碼簽署選項」)。您可以使用相同的憑證來簽署 AIR 檔案與安裝程式檔案，或是您也可以指定不同的憑證。當使用者從網路下載已簽署的 Windows Installer 檔案時，Windows 會根據憑證來識別該檔案的來源。

如需 -target 選項之外的 ADT 選項詳細資訊，請參閱 第 142 頁 「AIR Developer Tool (ADT)」。

下列範例會建立 DMG 檔 (Mac OS 適用的原生安裝程式檔案)：

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

下列範例會建立 EXE 檔 (Windows 適用的原生安裝程式檔案)：

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

下列範例會建立 EXE 檔並予以簽署：

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

建立使用原生擴充功能之應用程式的原生安裝程式

您可從來源檔案建立應用程式的原生安裝程式和應用程式所需的原生擴充功能套件。請使用 -package 命令和下列語法 (在單一命令列中)：

```
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 ...] ...
```

此語法與用於封裝原生安裝程式的語法相同，但是多兩個選項。使用 -extdir ***extension-directory*** 選項可指定包含應用程式所用 ANE 檔案 (原生擴充功能) 的目錄。使用選擇性 -migrate 旗標和 MIGRATION_SIGNING_OPTIONS 參數可在主要程式碼簽署憑證不同於先前版本所使用的憑證時，以移轉簽名簽署應用程式的更新。如需詳細資訊，請參閱第 169 頁「簽署 AIR 應用程式的更新版本」。

如需 ADT 選項的詳細資訊，請參閱 第 142 頁「AIR Developer Tool (ADT)」。

下列範例會建立使用原生擴充功能之應用程式的 DMG 檔案 (Mac OS X 的原生安裝檔案)。

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

從 **AIR** 檔案或 **AIRI** 檔案建立原生安裝程式

您可以使用 ADT，根據 AIR 檔案或 AIRI 檔案建立原生安裝程式檔案。若要根據 AIR 檔案建立原生安裝程式，請使用 ADT -package 命令和下列語法 (單行命令列)：

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

此語法和根據 AIR 應用程式來源檔案建立原生安裝程式的語法相似，不過有一些差異：

- 您可以指定 AIR 檔案做為來源，而不是指定 AIR 應用程式的應用程式描述器檔案與其他來源檔案。
- 不必為 AIR 檔案指定簽署選項，因為該檔案已經簽署

若要根據 AIRI 檔案建立原生安裝程式，請使用 ADT -package 命令和下列語法 (單行命令列)：

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

此語法和根據 AIR 檔案建立原生安裝程式的語法相似。不過有一些差異：

- 您可以指定 AIRI 檔案做為來源。
- 您需要為目標 AIR 應用程式指定簽署選項。

下列範例會根據 AIR 檔案建立 DMG 檔 (Mac OS 適用的原生安裝程式檔案)：

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

下列範例會根據 AIR 檔案建立 EXE 檔 (Windows 適用的原生安裝程式檔案)：

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

下列範例會建立 EXE 檔 (根據 AIR 檔案) 並予以簽署：

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

下列範例會根據 AIRI 檔案建立 DMG 檔 (Mac OS 適用的原生安裝程式檔案)：

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

下列範例會根據 AIRI 檔案建立 EXE 檔 (Windows 適用的原生安裝程式檔案)：

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

下列範例建立 EXE 檔案 (根據 AIRI 檔案) 並使用 AIR 與原生 Windows 簽名來簽署：

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

封裝桌上型電腦的固定執行階段組合包

固定執行階段組合包是包含應用程式程式碼，連同專用的執行階段版本。以這種方式封裝的應用程式會使用組合的執行階段，而非安裝於使用者電腦上其他位置的共用執行階段。

產生的組合包是 Windows 上應用程式檔案的獨立資料夾和 Mac OS 上的 .app 組合包。您必須產生在目標作業系統下執行之目標作業系統的組合包。(虛擬機器，例如 VMWare，可用於在一部電腦上執行多個作業系統)。

不需要安裝，即可從資料夾或組合包執行應用程式。

好處

- 產生獨立應用程式
- 安裝時不需要存取網際網路
- 從執行階段更新中隔離應用程式
- 企業可指定特定應用程式與執行階段的組合
- 支援傳統軟體部署模型
- 不需要重新散佈分開的執行階段
- 可使用 NativeProcess API
- 可使用原生擴充功能
- 可不受限制使用 File.openWithDefaultApplication() 函數。
- 不需要安裝，即可從 USB 或光碟機執行

缺點

- 當 Adobe 發佈安全性修補程式時，使用者無法自動取得重要的安全性修正
- 無法使用 .air 檔案格式
- 您必須建立自己的安裝程式 (如果有需要的話)
- 不支援 AIR 更新與 API 架構
- 不支援從網頁安裝與啟動 AIR 應用程式的 AIR 瀏覽器內建 API
- 在 Windows 中，必須使用您的安裝程式來組合檔案登錄
- 應用程式佔用的磁碟空間較大

在 Windows 上建立固定執行階段組合包

若要建立 Windows 的固定執行階段組合包，您必須在執行 Windows 作業系統時封裝應用程式。使用 ADT 組合包目標封裝應用程式：

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

此命令會在命名為 myApp 的的目錄中建立組合包。目錄包含您應用程式的檔案與執行階段檔案。您可直接從資料夾執行程式。但是，若要建立程式選單項目、登錄檔案類型或 URI 配置處理常式，您必須建立設定必要登錄項目的安裝程式。AIR SDK 不包含建立這類安裝程式的工具，但有一些可用的協力廠商選項，包含商業性與免費、開放來源安裝程式的工具套件。

您可在命令列 -target bundle 項目後指定第二組簽署選項，以在 WIndows 上簽署原生執行檔。這些簽署選項會識別私密金鑰與相關聯的憑證，以在套用原生 Windows 簽名時使用。(通常可使用 AIR 程式碼簽署憑證。) 僅會簽署主要的執行檔。此程序不會簽署以您應用程式封裝的任何其他執行檔。

檔案類型關聯

若要將您的應用程式與 Windows 公開或自訂的檔案類型相關聯，您的安裝程式必須設定適當的登錄項目。檔案類型也應該列於應用程式描述器檔案的 fileTypes 元素。

如需 Windows 檔案類型的詳細資訊，請參閱 MSDN Library: File Types and File Associations

URI 處理常式登錄

為了讓您的應用程式處理使用指定 URI 配置之 URL 的啟動，您的安裝程式必須設定必要的登錄項目。

如需登錄應用程式以處理 URI 配置的詳細資訊，請參閱 MSDN Library: Registering an Application to a URL Protocol

在 Mac OS X 上建立固定執行階段處理常式

若要建立 Mac OS X 的固定執行階段組合包，您必須在執行 Mac OS X 作業系統時封裝應用程式。使用 ADT 組合包目標封裝應用程式：

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

此命令會建立命名為 myApp.app 的應用程式組合包。組合包中包含您應用程式的檔案與執行階段檔案。您可按兩下 myApp.app 圖示來執行應用程式，並拖曳來安裝至適當位置，例如 Applications 資料夾。但是，若要登錄檔案類型或 URI 配置處理嘗試，您必須編輯應用程式套件內的屬性清單檔案。

散佈時，您可建立磁碟影像檔案 (.dmg)。Adobe AIR SDK 不提供建立固定執行階段組合包 dmg 檔案的工具。

檔案類型關聯

若要將您的應用程式與 Mac OS X 上公開或自訂的檔案類型相關聯，您必須編輯組合包中的 info.plist 檔案，以設定 CFBundleDocumentTypes 屬性。請參閱 Mac OS X Developer Library: Information Property List Key Reference, CFBundleURLTypes。

URI 處理常式登錄

為了讓您的應用程式處理使用指定配置的 URL 啟動，您必須編輯組合包中的 info.plist，以設定 CFBundleURLTypes 屬性。請參閱 Mac OS X Developer Library: Information Property List Key Reference, CFBundleDocumentTypes。

散佈桌上型電腦的 AIR 套件

AIR 應用程式可散佈為 AIR 套件，它包含了應用程式程式碼與所有資源。您可以透過任何一般方式散佈這個套件，例如提供下載、以電子郵件寄送，或是燒錄成 CD-ROM 等實體媒體。使用者只要按兩下 AIR 檔就能安裝應用程式。您可以使用 AIR 內部瀏覽器 API (Web 類型的 ActionScript 程式庫)，讓使用者按一下網頁上的單一連結，便能安裝您的 AIR 應用程式 (若有需要，一併安裝 Adobe® AIR®)。

AIR 應用程式也可封裝與散佈為原生安裝程式 (亦即，Windows 上的 EXE 檔案、Mac 上的 DMG 檔案，以及 Linux 上的 DEB 或 RPM 檔案)。原生安裝套件可根據相關平台慣例來散佈與安裝。散佈應用程式為原生套件時，會喪失 AIR 檔案格式的部分優點。亦即，單一安裝檔案將無法用於多數平台、無法再使用 AIR 更新架構，以及無法再使用內部瀏覽器 API。

在桌上型電腦上安裝及執行 AIR 應用程式

您只需要將 AIR 檔寄給收件者即可。例如，您可以隨電子郵件寄出 AIR 檔做為附件，或在網頁上提供其連結。

使用者下載 AIR 應用程式後，可依照下列指示進行安裝：

1 按兩下 AIR 檔。

使用者的電腦必須已經安裝 Adobe AIR。

2 在安裝視窗中，保持選取各項預設設定，然後按一下 「繼續」。

在 Windows 上，AIR 會自動執行下列事項：

- 將應用程式安裝於 Program Files 目錄
- 建立應用程式的桌面捷徑
- 建立「開始」選單捷徑
- 在「控制台」的「新增 / 移除程式」中加入應用程式項目

在 Mac OS 上，預設會將應用程式加入至 Applications 目錄。

如果先前已經安裝過該應用程式，安裝程式便會提供選項，讓使用者決定是要開啟現有版本的應用程式，或者更新成所下載 AIR 檔的版本。安裝程式會使用應用程式 ID 和 AIR 檔中的發行者 ID 來識別應用程式。

3 安裝完成後，按一下「完成」。

若要在 Mac OS 上安裝更新版的應用程式，使用者必須具有適當的系統權限，才能在應用程式目錄內進行安裝。在 Windows 和 Linux 上，使用者必須具有系統管理權限。

應用程式也可以透過 ActionScript 或 JavaScript 安裝新版本。如需詳細資訊，請參閱第 222 頁「更新 AIR 應用程式」。

一旦 AIR 應用程式安裝完成，使用者只要按兩下應用程式圖示即可開始執行，就如同任何其它桌面應用程式。

- 在 Windows 上，按兩下應用程式圖示 (安裝於桌面或特定資料夾)，或從「開始」選單選取應用程式。
- 在 Linux 上，按兩下應用程式圖示 (安裝於桌面或特定資料夾)，或從應用程式選單選取應用程式。
- 在 Mac OS 上，按兩下安裝資料夾中的應用程式。預設的安裝目錄為 /Applications 目錄。

備註：Linux 上只能安裝為 AIR 2.6 或更早版本開發的 AIR 應用程式。

AIR 的「隱藏安裝」功能讓使用者可藉由按一下網頁上的連結來安裝 AIR 應用程式。AIR 的「瀏覽器叫用」功能讓使用者可藉由按一下網頁上的連結來執行已安裝的 AIR 應用程式。下一節將說明這些功能。

從網頁安裝及執行桌上型 AIR 應用程式

AIR 內部瀏覽器 API 可讓您從網頁安裝與執行 AIR 應用程式。AIR 內部瀏覽器 API 是由 Adobe 掛載的 SWF 程式庫 air.swf 提供。AIR SDK 包括一個範例 “badge” 應用程式，它使用此程式庫來安裝、更新或啟動 AIR 應用程式 (若有需要，一併啟動執行階段)。您可以修改提供的範例 badge，或自行建立直接使用線上 air.swf 程式庫的 badge 網頁應用程式。

任何 AIR 應用程式皆可透過網頁 badge 來安裝。但是，只有在應用程式描述器檔案中包含 <allowBrowserInvocation>true</allowBrowserInvocation> 元素的應用程式，才可由網頁 badge 啟動。

更多說明主題

第 215 頁「AIR.SWF 內部瀏覽器 API」

桌上型電腦上的企業部署

IT 系統管理員可使用一般的桌面部署工具，以無訊息方式安裝 Adobe AIR 執行階段和 AIR 應用程式。IT 系統管理員可以執行下列作業：

- 使用 Microsoft SMS、IBM Tivoli 等工具，或任何能夠使用啟動載入器 (Bootstrapper) 進行無訊息安裝的部署工具，以無訊息方式安裝 Adobe AIR 執行階段
- 使用上述的執行階段部署工具，以無訊息方式安裝 AIR 應用程式

如需詳細資訊，請參閱「Adobe AIR Administrator's Guide」(http://www.adobe.com/go/learn_air_admin_guide_tw)。

安裝記錄 (桌上型電腦)

安裝 AIR 執行階段本身或 AIR 應用程式時，都會記錄安裝記錄。您可以檢查記錄檔，判斷任何安裝或更新問題的發生原因。

記錄檔會在下列位置建立：

- Mac：標準系統記錄 (/private/var/log/system.log)

  您可以開啟 「主控台」應用程式 (通常位於 「公用程式」資料夾)，以檢視 Mac 系統記錄。
- Windows XP：C:\Documents and Settings\< 使用者名稱 >\Local Settings\Application Data\Adobe\AIR\logs\Install.log
- Windows Vista、Windows 7：C:\Users\< 使用者名稱 >\AppData\Local\Adobe\AIR\logs\Install.log
- Linux：/home/< 使用者名稱 >/.appdata/Adobe/AIR/Logs/Install.log

備註：AIR 2 以前的 AIR 版本不會建立這些記錄檔。

第 7 章 開發行動裝置 AIR 應用程式

行動裝置上的 AIR 應用程式會部署為原生應用程式。它們將使用裝置的應用程式格式，而非 AIR 檔案格式。AIR 目前支援 Android APK 套件與 iOS IPA 套件。建立應用程式套件的發佈版本後，便可經由標準平台機制來散佈您的應用程式。針對 Android，這通常是指 Android Market；針對 iOS，則是指 Apple App Store。

您可以使用 AIR SDK 與 Flash Professional、Flash Builder 或其他 ActionScript 開發工具，以建立行動裝置的 AIR 應用程式。目前不支援 HTML 類型的行動裝置 AIR 應用程式。

備註：Research In Motion (RIM) BlackBerry Playbook 提供自己的 SDK 做為 AIR 開發之用。如需 Playbook 開發的詳細資訊，請參閱 RIM: BlackBerry Tablet OS Development。

備註：本文件說明如何使用 AIR 2.6 SDK 或更新版本開發 iOS 應用程式。以 AIR 2.6 以上版本建立的應用程式可以安裝在執行 iOS 4 或更新版本的 iPhone 3Gs、iPhone 4 和 iPad 裝置上。若要開發適用於較早版本的 iOS 的 AIR 應用程式，您必須使用 AIR 2 Packager for iPhone，如建立 iPhone 應用程式中所描述。

如需有關隱私權最佳作法的詳細資訊，請參閱 Adobe AIR SDK 隱私權指南。

如需執行 AIR 應用程式的完整系統需求，請參閱 Adobe AIR 系統需求。

設定您的開發環境

行動平台會有額外的設定需求，超出一般 AIR、Flex 與 Flash 開發環境的設定。(如需設定基本 AIR 開發環境的詳細資訊，請參閱 第 15 頁 「AIR 開發適用的 Adobe Flash Platform 工具」)。

Android 設定

AIR 2.6+ 中的 Android 通常不需要進行特殊設定。Android ADB 工具包含在 AIR SDK 中 (位於 lib/android/bin 資料夾)。AIR SDK 使用 ADB 工具，在裝置上安裝、解除安裝與執行應用程式套件。您也可以使用 ADB 檢視系統記錄。若要建立與執行 Android 模擬器，您必須另外下載 Android SDK。

如果您的應用程式在應用程式描述器的 <manifestAdditions> 元素中加入目前版本的 AIR 未視為有效的元素，您必須安裝更新版本的 Android SDK。請將 AIR_ANDROID_SDK_HOME 環境變數或 -platformsdk 命令列參數設定為 SDK 的檔案路徑。AIR 封裝工具 ADT 會使用這個 SDK 驗證 <manifestAdditions> 元素中的項目。

在 AIR 2.5 中，您必須從 Google 下載另一個 Android SDK 複本。您可以設定 AIR_ANDROID_SDK_HOME 環境變數，以參照 Android SDK 資料夾。若未設定此環境變數，您必須在 ADT 命令列的 -platformsdk 引數中指定 Android SDK 的路徑。

更多說明主題

第 160 頁 「ADT 環境變數」

第 263 頁 「Path 環境變數」

iOS 設定

若要在裝置上安裝與測試 iOS 應用程式與散佈該應用程式，您必須加入 Apple iOS Developer 計劃 (為付費服務)。在您加入 iOS Developer 計劃後，即可存取 iOS Provisioning Portal，您可以在此從 Apple 取得下列項目與檔案，這些項目與檔案將用以在裝置上安裝應用程式，以供測試和後續散佈之用。這些項目與檔案包括：

- 開發與發佈憑證

- 應用程式 ID
- 開發與發佈佈建檔案

行動應用程式設計注意事項

撰寫程式碼與設計時，必須注意行動裝置的操作環境與實體特性。例如，精簡程式碼以儘量加快執行速度，便是需要注意的重要事項。當然，程式碼最佳化有其極限；聰明的設計可以在裝置的限制內，讓視覺呈現不會造成顯示系統的過份負擔。

程式碼

提高程式碼執行速度總是有所助益，多數行動裝置的處理器速度較慢，因此更值得花費時間撰寫精簡的程式碼。此外，行動裝置大部分都是仰賴電池。以較少的工作量達成相同的結果，將可節省電池電力。

設計

設計應用程式的使用者體驗時，應考量許多因素，像是螢幕較小、觸控螢幕的互動模式，甚至是行動使用者不斷變化的環境等等。

同時考量程式碼與設計

如果應用程式使用動畫，顯示最佳化便十分重要。不過，單是程式碼最佳化通常尚且不足。您必須設計應用程式的視覺外觀，以便讓程式碼可以更有效率地顯示。

最佳化 Flash Platform 效能指南中討論重要的最佳化技術。指南中涵蓋的技術適用於所有 Flash 與 AIR 內容，對於開發可在行動裝置上順利執行的應用程式也十分重要。

- Paul Trani：行動 Flash 開發的提示與祕訣 (英文)
- roguish：行動裝置版 AIR 的 GPU 測試應用程式
- Jonathan Campos：AIR for Android 應用程式的最佳化技術
- Charles Schulze：AIR 2.6 遊戲開發：內含 iOS (英文)

應用程式效期

當焦點從應用程式移至其他應用程式時，AIR 會將影格速率降低為每秒 4 個影格並停止顯示圖示。若低於此影格速率，串流網路與 Socket 連線將可能中斷。若應用程式未使用此類連線，可以調整更低的影格速率。

如果合適，應停止音效播放，並移除地理位置與加速計感應器的偵聽程式。AIR NativeApplication 物件會傳送 activate 與 deactivate 事件。使用這些事件可管理作用中與背景狀態的轉換。

多數行動作業系統會在沒有預警的情況下終止背景應用程式。藉由經常儲存應用程式的狀態，不論應用程式從背景返回作用中狀態或是重新啟動，都可將應用程式還原至可接受的狀態。

資訊密度

行動裝置螢幕的實際大小小於桌上型電腦，但行動裝置螢幕的像素密度 (每英吋像素) 較高。若使用相同的字體大小，行動裝置上的字母實體大小會小於桌上型電腦上的字母。通常必須使用較大的字體，以確保使用者能夠閱讀。一般而言，14 點是可以輕鬆閱讀的最小字體大小。

您通常會在移動及光線不足的情況下使用行動裝置。請考慮螢幕實際能顯示的資訊數量。與相同像素尺寸的桌上型電腦螢幕相比，行動裝置所能顯示的資訊量可能較少。

同時也考量使用者觸控螢幕時，手指與手會遮住部分顯示。當使用者必須以較長時間觸控使用互動元素時，請將這些元素置於螢幕的兩側與底部。

文字輸入

許多裝置使用虛擬鍵盤供文字輸入。虛擬鍵盤會遮住部分螢幕，且通常難以使用。避免仰賴鍵盤事件 (軟鍵除外)。

考慮使用其他方法，而不使用輸入文字欄位。例如，若要讓使用者輸入數字，則不需文字欄位。您可以提供兩個按鈕來增加或減少數值。

軟鍵

行動裝置包含數量不一的軟鍵。軟鍵為可由程式控制不同功能的按鈕。在應用程式中使用這些按鍵，請遵循平台的慣例。

螢幕方向變更

行動內容可以直向或橫向檢視。請考量您的應用程式將如何處理螢幕方向變更。如需詳細資訊，請參閱舞台方向。

螢幕變暗

顯示視訊時，AIR 不會自動避免螢幕變暗。您可以使用 AIR NativeApplication 物件的 systemIdleMode 屬性，控制裝置是否進入省電模式。(在某些平台上，必須要求適當權限，才能執行此功能)。

來電

當使用者撥打或接聽電話時，AIR 執行階段會自動將音訊靜音。在 Android 上，如果應用程式在背景播放音效，應該設定應用程式描述器中的 Android READ_PHONE_STATE 權限。否則，Android 不會讓執行階段偵測通話並自動將音效靜音。請參閱 第 66 頁 「Android 權限」。

感應區目標

設計使用者可以點選的按鈕及其他使用者介面元素時，請考量感應區目標的大小。這些元素的大小，必須足以讓手指在觸控螢幕上輕易的啟動它們。此外，您也必須確定各個目標之間有足夠的空間。在一般高解析度手機螢幕上，感應區目標區域每邊應大約在 44 像素至 57 像素。

應用程式套件安裝大小

行動裝置安裝應用程式與資料的儲存空間通常遠小於桌上型電腦。移除未使用的資源與程式庫，可最小化套件大小。

在 Android 上，安裝應用程式時，應用程式套件不會解壓縮為個別的檔案。而是在存取資源時，將資源解壓縮至暫時儲存空間。為了最小化此解壓縮資源儲存佔用空間，當完全載入資源時，請關閉檔案與 URL 串流。

檔案系統存取

不同的行動作業系統會實施不同的檔案系統限制，而這些限制通常與桌上型作業系統實施的限制不同。因此，適當的檔案和資料儲存位置也會因平台而不同。

檔案系統不同會造成 AIR File 類別提供的常見目錄捷徑不一定可用。下表顯示可在 Android 和 iOS 上使用的捷徑：

	Android	iOS
File.applicationDirectory	透過 URL 唯讀 (非原生路徑)	唯讀
File.applicationStorageDirectory	可用	可用
File.cacheDirectory	可用	可用

	Android	iOS
File.desktopDirectory	sdcard 的根目錄	無法使用
File.documentsDirectory	sdcard 的根目錄	可用
File.userDirectory	sdcard 的根目錄	無法使用
File.createTempDirectory()	可用	可用
File.createTempFile()	可用	可用

iOS 應用程式的 Apple 原則針對檔案在不同情況應使用的儲存位置，提供特定規則。例如，有個原則是，若檔案包含使用者所輸入的資料或包含無法以其他方式重新產生或下載的資料，則只能儲存在指定做為遠端備份之用的目錄中。如需有關如何符合 Apple 檔案備份和快取原則的資訊，請參閱控制檔案備份和快取。

UI 組件

Adobe 已經開發 Flex 架構的行動裝置最佳化版本。如需詳細資訊，請參閱使用 Flex 和 Flash Builder 開發行動應用程式。

此外，也有適用於行動應用程式的社群組件專案。包括：

- Josh Tynjala 的適用於 Starling 的 Feathers UI 控制項
- Derrick Grigg 的 skinnable version of Minimal Comps (極小運算的可設定外觀版本)
- Todd Anderson 的 as3flobile 組件

Stage 3D 加速圖形顯示方式

從 AIR 3.2 開始，行動裝置版 AIR 支援 Stage 3D 加速圖形顯示方式。Stage3D ActionScript API 是一組啟用進階 2D 和 3D 功能的低階 GPU 加速 API。這些低階 API 讓開發人員可以彈性地利用 GPU 硬體加速，顯著改善效能。此外，您也可以使用支援 Stage3D ActionScript API 的遊戲引擎。

如需詳細資訊，請參閱 Gaming engines, 3D, and Stage 3D (遊戲引擎、3D 和 Stage 3D)。

視訊平滑化

為了提高效能，AIR 上的視訊平滑化功能已停用。

原生功能

AIR 3.0+

許多行動平台均提供尚無法透過標準 AIR API 存取的功能。從 AIR 3 開始，您可以使用自己的原生程式碼元件庫來擴充 AIR。這些原生擴充功能元件庫可以存取作業系統中的功能，甚至是特別為指定之裝置所提供的功能。原生擴充功能可以使用 iOS 上的 C 以及 Android 上的 Java 或 C 來撰寫。如需有關開發原生擴充功能的詳細資訊，請參閱 Adobe AIR 原生擴充功能簡介。

建立行動裝置 AIR 應用程式的工作流程

一般而言，建立行動 (或其他) 裝置 AIR 應用程式的工作流程，相當類似於建立桌上型應用程式。主要的工作流程差異在於封裝、除錯與安裝應用程式的時候。例如，AIR for Android 應用程式使用原生 Android APK 套件格式，而非 AIR 套件格式。因此，它們也使用標準的 Android 安裝與更新機制。

AIR for Android

開發 Android 的 AIR 應用程式時，以下為典型的步驟：

- 撰寫 ActionScript 或 MXML 程式碼。
- 建立 AIR 應用程式描述器檔案 (使用 2.5 或更新的命名空間)。
- 編譯應用程式。
- 將應用程式封裝為 Android 套件 (.apk)。
- 在裝置或 Android 模擬器 (如果使用外部執行階段) 上安裝 AIR 執行階段 (在 AIR 3.7 及更新版本中，預設為固定執行階段)。
- 在裝置 (或 Android 模擬器) 上安裝應用程式。
- 在裝置上啟動應用程式。

您可以使用 Adobe Flash Builder、Adobe Flash Professional CS5，或命令列工具來完成這些步驟。

完成 AIR 應用程式且封裝為 APK 檔案後，可將它提交至 Android Market 或經由其他方式散佈。

AIR for iOS

開發 iOS 的 AIR 應用程式時，以下為典型的步驟：

- 安裝 iTunes。
- 在 Apple iOS Provisioning Portal 上，產生所需的開發人員檔案與 ID。這些項目包括：
  - 開發人員憑證
  - 應用程式 ID
  - 佈建描述檔

  建立佈建描述檔時，必須列出計畫安裝應用程式的任何測試裝置的 ID。
- 將開發憑證與私密金鑰轉換為 P12 金鑰儲存檔案。
- 撰寫應用程式 ActionScript 或 MXML 程式碼。
- 使用 ActionScript 或 MXML 編譯器編譯應用程式。
- 建立應用程式的圖示圖案及起始螢幕圖案。
- 建立應用程式描述器 (使用 2.6 或更新的命名空間)。
- 使用 ADT 封裝 IPA 檔案。
- 使用 iTunes 將佈建描述檔置於測試裝置上。
- 在 iOS 裝置上安裝及測試應用程式。您可以使用 iTunes 或透過 USB 使用 ADT (AIR 3.4 及以上版本支援 USB) 來安裝 IPA 檔案。

完成 AIR 應用程式後，可使用發佈憑證與佈建描述檔來重新封裝。之後，便可將它提交至 Apple App Store。

設定行動應用程式屬性

如同其他 AIR 應用程式，設定應用程式描述器檔案中的基本應用程式屬性。行動應用程式會忽略某些桌上型特定的屬性，像是視窗大小與透明度。行動應用程式也可使用其平台特定的屬性。例如，Android 應用程式可以包含 android 元素，iOS 應用程式可以包含 iPhone 元素。

一般設定

對於所有行動裝置應用程式，有數個重要的應用程式描述器設定。

必要的 AIR 執行階段版本

使用應用程式描述器檔案的命名空間，指定應用程式所需的 AIR 執行階段版本。

application 元素中指派的命名空間，大致上決定了應用程式可使用的功能。例如，如果應用程式使用 AIR 2.7 命名空間，而使用者安裝了較新的版本，則應用程式仍然會出現 AIR 2.7 的行為 (即使在較新版本中已變更該行為)。只有當您變更命名空間且發佈更新，應用程式才能存取新行為與功能。安全性修正為此規則的重要例外。

在分開使用執行階段與應用程式的裝置上 (像是 AIR 3.6 和更早版本上的 Android)，若使用者沒有所需的版本，將會收到安裝或升級 AIR 的提示。在整合執行階段的裝置上 (像是 iPhone)，將不會發生此狀況 (因為需要的版本一開始便與應用程式一起封裝)。

備註：(AIR 3.7 及更新版本) 根據預設，ADT 會將執行階段與 Android 應用程式一起封裝。

使用根 application 元素的 xmlns 特質，指定命名空間。下列命名空間可用於行動應用程式 (取決於目標行動平台)：

```
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">
```

備註：以 AIR 2.0 SDK 為基礎，iPhone SDK 的封裝程式提供 iOS 3 裝置的支援。如需針對 iOS 3 建立 AIR 應用程式的詳細資訊，請參閱建立 iPhone 應用程式。AIR 2.6 SDK (及更新的版本) 支援 iOS 4，以及其上的 iPhone 3Gs、iPhone 4 和 iPad 裝置。

更多說明主題

第 178 頁「application」

應用程式身分識別

針對發佈的每個應用程式，有數種設定必須是唯一的。這些設定包括 ID、名稱與檔案名稱。

Android 應用程式 ID

在 Android 上，ID 會轉換為 Android 套件名稱，方法是在 AIR ID 之前加上 “air.”。因此，如果 AIR ID 為 com.example.MyApp，則 Android 套件名稱為 air.com.example.MyApp。

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

此外，如果 ID 不是 Android 作業系統上合法的套件名稱，則會轉換為合法的名稱。連字號字元會變更為底線，任何 ID 組件中的前導數字前會加上大寫 “A”。例如，ID：3-goats.1-boat 會轉變為套件名稱：air.A3_goats.A1_boat。

備註：新增至應用程式 ID 的前置詞，可用來在 Android Market 中識別 AIR 應用程式。如果不要讓應用程式因前置詞而識別為 AIR 應用程式，必須如 退出 Android 的 AIR 應用程式分析所述，解除封裝 APK 檔案、變更應用程式 ID，並將它重新封裝。

iOS 應用程式 ID

設定 AIR 應用程式 ID 以符合您在 Apple iOS Provisioning Portal 中建立的應用程式 ID。

iOS App ID 包含一個組合包種子 ID，隨後為組合包識別名稱。組合種子 ID 是 Apple 指派給應用程式 ID 的一個字元字串，例如 5RM86Z4DJM。組合包識別名稱是您所選擇的反向網域樣式的名稱。組合包識別名稱可能以星號 (*) 結尾，這表示萬用字元應用程式 ID。如果組合包識別名稱以萬用字元結尾，您可以使用任何合法字串來取代該萬用字元。

例如：

- 如果 Apple 應用程式 ID 為 5RM86Z4DJM.com.example.helloWorld，您必須在應用程式描述器中使用 com.example.helloWorld。
- 如果 Apple 應用程式 ID 為 96LPVWEASL.com.example.* (萬用字元應用程式 ID)，則您可使用 com.example.helloWorld 或 com.example.anotherApp，或以 com.example 開頭的其他 ID。
- 最後，如果 Apple 應用程式 ID 僅為組合包種子 ID 與一個萬用字元，像是：38JE93KJL.*，則您可以使用 AIR 中的任何應用程式 ID。

指定應用程式 ID 時，請不要包含應用程式 ID 的組合包種子 ID 部分。

更多說明主題

第 193 頁「id」

第 188 頁「filename」

第 200 頁「name」

應用程式版本

在 AIR 2.5 與更新的版本中，在 versionNumber 元素中指定應用程式版本。version 元素已不再可供使用。指定 versionNumber 的值時，必須使用最多三個號碼的序列，號碼之間以點分隔，例如：“0.1.2”。版本號碼的每個區段最多可以有三位數字。(亦即，“999.999.999” 為允許的最大版本號碼)。號碼中毋須包括所有三個區段；“1” 與 “1.0” 都是合法的版本號碼。

您也可以使用 versionLabel 元素，指定版本的標籤。新增版本標籤時，在 Android 應用程式資訊畫面等處，會顯示標籤以取代版本號碼。使用 Android Market 散佈的應用程式，必須指定版本標籤。如果在 AIR 應用程式描述器中未指定 versionLabel 值，則 versionNumber 值會指派給 Android 版本標籤欄位。

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

在 Android 上，AIR versionNumber 會轉譯為 Android 整數 versionCode，使用的公式為：a*1000000 + b*1000 + c，其中 a、b 及 c 是 AIR 版本號碼 a.b.c 的各組件。

更多說明主題

第 207 頁「version」

第 208 頁「versionLabel」

第 208 頁「versionNumber」

主要應用程式 SWF

在 initialWindow 元素的 content 子元素中，指定主要應用程式 SWF 檔案。以行動描述檔中的裝置為目標時，必須使用 SWF 檔案 (不支援 HTML 類型應用程式)。

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

您必須將檔案包括在 AIR 套件中 (使用 ADT 或您的 IDE)。只是參照應用程式描述器中的名稱，不會將檔案自動包括在套件中。

主要螢幕屬性

initialWindow 元素中的數個子元素，控制主要應用程式螢幕的初始外觀與行為。

- **aspectRatio** — 指定應用程式一開始應顯示為 portrait 格式 (高度大於寬度)、landscape 格式 (高度小於寬度) 或 any 格式 (舞台會自動朝向所有方向)。

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

- **autoOrients** — 指定使用者旋轉裝置或執行其他與方向相關的手勢時 (例如，開啟或關閉滑蓋式鍵盤)，是否自動變更舞台方向。若設為 false (預設)，則舞台不會隨著裝置變更方向。

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

- **depthAndStencil** — 指定使用深度或模板緩衝區。您通常會在使用 3D 內容時使用這些緩衝區。

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

- **fullScreen** — 指定應用程式是否應佔用整個裝置顯示，或是與一般作業系統顏色 (像是系統狀態列) 共用顯示。

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

- **renderMode** — 指定執行階段應以圖像處理單元 (GPU) 或主要的中央處理單元 (CPU) 來顯示應用程式。一般而言，GPU 顯示可提高顯示速度，但有些功能 (例如，某些混合模式與 PixelBender 濾鏡) 在 GPU 模式中無法使用。此外，不同的裝置與不同的裝置驅動程式，會有不同的 GPU 功能與限制。您應該儘可能在多種裝置上測試應用程式，特別是使用 GPU 模式的時候。

  您可以設定顯示模式為 gpu、cpu、direct 或 auto。預設值為 auto，它目前會退回 cpu 模式。

  備註：若要在行動平台上使用 AIR 來利用 Flash 內容的 GPU 加速，Adobe 建議您使用 renderMode="direct" (即 Stage3D)，而不要使用 renderMode="gpu"。Adobe 正式支援並建議使用下列 Stage3D 架構：Starling (2D) 和 Away3D (3D)。如需有關 Stage3D 和 Starling/Away3D 的詳細資訊，請參閱 http://gaming.adobe.com/getstarted/。

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

  備註：在背景執行的應用程式無法使用 renderMode=”direct”。

  GPU 模式的限制如下：

  - Flex 架構不支援 GPU 顯示模式。
  - 不支援濾鏡
  - 不支援 PixelBender 混合與填色
  - 不支援下列混合模式：圖層、Alpha、清除、重疊、實光、變亮與變暗
  - 不建議在播放視訊的應用程式中使用 GPU 顯示模式。
  - 在 GPU 顯示模式中，當虛擬鍵盤開啟時，文字欄位並不會適當地移到可視的位置。若要確保使用者在輸入文字時看得見您的文字欄位，請使用舞台的 softKeyboardRect 屬性以及軟體鍵盤事件，將文字欄位移至可見區域。
  - 如果 GPU 無法顯示某個顯示物件，則完全不會顯示。例如，如果套用濾鏡至顯示物件，則不會顯示該物件。

  備註：在 AIR 2.6+ 中，iOS 的 GPU 實作與先前 AIR 2.0 版本中的實作差異頗大。兩者使用不同的最佳化考量。

更多說明主題

第 181 頁「aspectRatio」

第 181 頁「autoOrients」

第 184 頁「depthAndStencil」

第 192 頁「fullScreen」

第 202 頁「renderMode」

支援的描述檔

您可以新增 supportedProfiles 元素，指定應用程式支援的裝置描述檔。使用 mobileDevice 描述檔於行動裝置。使用 Adobe Debug Launcher (ADL) 執行應用程式時，ADL 會使用清單中的第一個描述檔作為作用中描述檔。執行 ADL 時也可使用 -profile 旗標，在支援的清單中選取特定的描述檔。如果應用程式可在所有描述檔之下執行，您可以完全不包括 supportedProfiles 元素。在此情況下，ADL 使用桌上型描述檔作為預設的作用中描述檔。

若要指定應用程式支援行動裝置與桌上型描述檔，且通常要在行動描述檔中測試應用程式，請新增下列元素：

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

更多說明主題

第 205 頁 「supportedProfiles」

第 211 頁 「裝置描述檔」

第 137 頁 「AIR Debug Launcher (ADL)」

必須要有原生擴充功能

支援 mobileDevice 描述檔的應用程式可以使用原生擴充功能。

請在應用程式描述器中宣告 AIR 應用程式會使用的所有原生擴充功能。下列範例說明指定兩個必要原生擴充功能的語法：

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

extensionID 元素的值與擴充功能描述器檔案中的 id 元素相同。擴充功能描述器檔案是一個名為 extension.xml 的 XML 檔。它是封裝於您從原生擴充功能開發人員得到的 ANE 檔案。

虛擬鍵盤行為

若要將執行階段為了維護焦點文字輸入欄位的可見狀態，對升起的虛擬鍵盤進行自動平移和調整大小的行為變成停用狀態，請將 softKeyboardBehavior 元素設為 none。如果停用自動行為，則應用程式應確認升起虛擬鍵盤時，可見到文字輸入區域或其他相關內容。您可以使用舞台的 softKeyboardRect 屬性與 SoftKeyboardEvent，偵測鍵盤開啟並判斷它遮住的區域。

若要啟用自動行為，請將元素值設為 pan：

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

由於 pan 為預設值，忽略 softKeyboardBehavior 元素也可啟用自動鍵盤行為。

備註：當您也使用 GPU 顯示時，不支援平移行為。

更多說明主題

第 204 頁 「softKeyboardBehavior」

Stage.softKeyboardRect

SoftKeyboardEvent

Android 設定

在 Android 平台上，您可以使用應用程式描述器的 android 元素，新增資訊至 Android 應用程式資訊清單，這是 Android 作業系統所使用的應用程式屬性檔案。建立 APK 套件時，ADT 會自動產生 Android Manifest.xml 檔案。AIR 會將某些屬性設定為特定功能運作所需的值。在 AIR 應用程式描述器 android 區段中設定的任何其他屬性，會新增至 Manifest.xml 檔案中對應的區段。

備註：對於多數 AIR 應用程式，您必須在 android 元素中設定應用程式所需的 Android 權限，但是通常不需要設定任何其他屬性。

您只能設定接受字串、整數或布林值的特質。不支援在應用程式套件中設定資源參照。

備註：執行階段至少需要版本 14 以上的 SDK。如果您希望建立僅適用於較高版本的應用程式，請確定資訊清單包含正確版本的相應 <uses-sdk android:minSdkVersion=""></uses-sdk>。

保留的 Android 資訊清單設定

AIR 會在產生的 Android 資訊清單文件中設定數個資訊清單項目，以確保應用程式與執行階段功能可正常運作。您無法定義下列設定：

manifest 元素

您無法設定 manifest 元素的下列特質：

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

activity 元素

您無法設定主要 activity 元素的下列特質：

- android:label
- android:icon

application 元素

您無法設定 application 元素的下列特質：

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

Android 權限

Android 安全模型規定每個應用程式必須要求權限，才能使用涉及安全或隱私的功能。這些權限必須在封裝應用程式時指定，且無法在執行階段變更。當使用者安裝應用程式時，Android 作業系統會通知使用者應用程式所要求的權限。如果未要求某個功能所需的權限，當應用程式存取該功能時，Android 作業系統可能會擲出例外，但不保證一定會有例外。執行階段會將例外傳輸至您的應用程式。若出現無訊息失敗的狀況，會將權限失敗訊息新增至 Android 系統記錄。

在 AIR 中，必須在應用程式描述器的 android 元素內指定 Android 權限。將使用下列格式新增權限 (其中的 PERMISSION_NAME 為 Android 權限的名稱)：

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

manifest 元素內的 uses-permissions 陳述式會直接新增至 Android 資訊清單文件。

使用各種 AIR 功能需要下列權限：

ACCESS_COARSE_LOCATION 允許應用程式透過 Geolocation 類別存取 WIFI 與行動電話網路位置資料。

ACCESS_FINE_LOCATION 允許應用程式透過 Geolocation 類別存取 GPS 資料。

ACCESS_NETWORK_STATE 與 ACCESS_WIFI_STATE 允許應用程式透過 NetworkInfo 類別存取網路資訊。

CAMERA 允許應用程式存取攝影機。

備註：當您要求使用攝影機功能的權限時，Android 會假設您的應用程式也需要攝影機。如果攝影機為應用程式的選擇性功能，則應為攝影機新增 uses-feature 元素至資訊清單，將所需的特質設為 false。請參閱 第 68 頁 「Android 相容性篩選」。

INTERNET 允許應用程式提出網路要求。也允許遠端除錯。

READ_PHONE_STATE 允許 AIR 執行階段在通話期間將音效靜音。如果應用程式在背景播放音效，則應設定此權限。

RECORD_AUDIO 允許應用程式存取麥克風。

WAKE_LOCK 與 DISABLE_KEYGUARD 允許應用程式使用 SystemIdleMode 類別設定，避免裝置進入休眠。

WRITE_EXTERNAL_STORAGE 允許應用程式寫入裝置上的外部記憶卡。

例如，若要設定需要每個權限的應用程式的權限，可以新增下列項目至應用程式描述器：

```
<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>
```

更多說明主題

Android 安全與權限

Android Manifest.permission 類別

Android 自訂 URI 配置

您可以使用自訂 URI 配置，從網頁或原生 Android 應用程式來啟動 AIR 應用程式。自訂 URI 支援有賴於 Android 資訊清單中指定的比對方式篩選，因此此技術無法用於其他平台。

若要使用自訂 URI，請新增 intent-filter 至 <android> 區塊內的應用程式描述器。下列範例中的兩個 intent-filter 元素都必須指定。編輯 <data android:scheme="***my-customuri***"/> 陳述式，以反映自訂配置的 URI 字串。

```
<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>
```

比對方式篩選會通知 Android 作業系統，應用程式已經可以執行指定的動作。在自訂 URI 的狀況下，這代表使用者按一下使用該 URI 配置的連結 (且瀏覽器不知道如何處理)。

當經由自訂 URI 叫用應用程式時，NativeApplication 物件會傳送 invoke 事件。包括查詢參數的連結 URL，置於 InvokeEvent 物件的 arguments 陣列中。您可以使用任何數量的 intent-filters。

備註：StageWebView 實體中的連結無法開啟使用自訂 URI 配置的 URL。

更多說明主題

Android 比對方式篩選

Android 動作與類別

Android 相容性篩選

Android 作業系統使用應用程式資訊清單檔案中的數個元素，判斷應用程式是否與指定的裝置相容。新增此資訊至資訊清單為選擇性。若未包含這些元素，應用程式將可安裝在任何 Android 裝置上。不過，應用程式不見得可以在任何 Android 裝置上正確地運作。例如，在沒有攝影機的手機上，攝影機應用程式用途不大。

可用來篩選的 Android 資訊清單標記包括：

- supports-screens
- uses-configuration
- uses-feature
- uses-sdk (在 AIR 3+ 中)

攝影機應用程式

如果為應用程式要求攝影機權限，Android 會假設應用程式需要所有可用的攝影機功能，包括自動對焦與閃光燈。如果應用程式不需要所有的攝影機功能，或攝影機為選擇性功能，您應該設定攝影機各個 uses-feature 元素，指示選擇性的功能。否則，當使用者的裝置缺乏其中一項功能或完全沒有攝影機時，將無法在 Android Market 上找到您的應用程式。

下列範例說明如何要求攝影機權限，並讓所有攝影機功能成為選擇性：

```
<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>
```

音效錄製應用程式

如果要求音效錄製的權限，Android 也會假設應用程式需要麥克風。如果音效錄製為應用程式選擇性功能，可以新增 uses-feature 標記，指定麥克風並非必要。否則，當使用者的裝置沒有麥克風時，將無法在 Android Market 上找到您的應用程式。

下列範例說明如何要求使用麥克風的權限，並將麥克風硬體設為選擇性：

```
<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>
```

更多說明主題

Android 開發人員：Android 相容性

Android 開發人員：Android 功能名稱常數

安裝位置

您可以將 Android manifest 元素的 installLocation 特質設為 auto 或 preferExternal，以允許應用程式安裝或移至外部記憶卡：

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

Android 作業系統不保證您的應用程式將會安裝至外部記憶體。使用者也可以使用系統設定應用程式，在內部與外部記憶體之間移動應用程式。

應用程式即使安裝在外部記憶體，應用程式的快取與使用者資料 (像是 app-storage 目錄的內容、共用的物件與暫存檔案) 仍然儲存在內部記憶體中。為了避免使用過多的內部記憶體，請慎選儲存至應用程式儲存目錄的資料。大量資料應使用 File.userDirectory 或 File.documentsDirectory 位置 (兩者皆對應至 Android 上 SD 卡的根目錄) 儲存至 SDCard。

在 StageWebView 物件中啟用 Flash Player 和其他外掛程式

在 Android 3.0+ 中，應用程式必須在 Android 應用程式元素中啟用硬體加速，外掛程式內容才能在 StageWebView 物件中顯示。若要讓外掛程式顯示，請將 application 元素的 android:hardwareAccelerated 屬性設定為 true：

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

顏色深度

AIR 3+

在 AIR 3 與更新的版本中，執行階段會設定顯示，以顯示 32 位元色彩。在舊版 AIR，執行階段會使用 16 位元色彩。您可以使用應用程式描述器的 <colorDepth> 元素，指示執行階段使用 16 位元色彩：

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

使用 16 位元顏色深度可提高顯示效能，但會　牲顏色精確度。

iOS 設定

在應用程式描述器中，僅能套用至 iOS 裝置的設定會置於 <iPhone> 元素內。iPhone 元素可以具有 InfoAdditions 元素、requestedDisplayResolution 元素、Entitlements 元素、externalSwfs 元素和 forceCPURenderModeForDevices 做為子系。

InfoAdditions 元素可讓您指定索引鍵值配對，以新增至應用程式的 Info.plist 設定檔案。例如，下列值會設定應用程式的狀態列樣式，並指出應用程式不需要持續存取 Wi-Fi。

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

InfoAdditions 設定包括在 CDATA 標籤中。

Entitlements 元素可讓您指定索引鍵值配對，以新增至應用程式的 Entitlements.plist 設定檔案。Entitlements.plist 設定能讓應用程式存取特定 iOS 功能，例如推送通知。

如需 Info.plist 和 Entitlements.plist 設定的詳細資訊，請參閱 Apple 開發人員文件。

在 iOS 上支援背景工作

AIR 3.3

Adobe AIR 3.3 及更新版本可啟用特定背景行為，以便在 iOS 上支援多工作業：

- 音效
- 位置更新
- 網路
- 選擇結束背景應用程式執行

備註：在 SWF 版本 21 及舊版中，當設定 「直接」顯示模式時，AIR 並未支援背景執行。由於這個限制，Stage3D 應用程式無法執行背景工作，例如音效播放、位置更新、網路上傳或下載等。iOS 不允許在背景執行 OpenGLES 或顯示呼叫。嘗試在背景進行 OpenGL 呼叫的應用程式會遭到 iOS 加以終止。Android 並未限制應用程式在背景進行 OpenGLES 呼叫或是執行其他背景工作 (如音效播放)。透過 SWF 版本 22 及更新版本，AIR 行動裝置應用程式可以在已設定 renderMode direct 的情況下在背景執行。如果在背景進行 OpenGLES 呼叫，AIR iOS 執行階段會造成 ActionScript 錯誤 (3768 - 在背景執行期間不會使用 Stage3D API)。不過，在 Android 則不會造成任何錯誤，因為 Android 原生應用程式是可以在背景進行 OpenGLES 呼叫的。為了讓行動裝置資源的運用達到最佳化，當應用程式在背景執行時，請不要進行顯示呼叫。

背景音效

若要啟用背景音效播放和錄製，請在 InfoAdditions 元素中加入下列索引鍵值配對：

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

背景位置更新

若要啟用背景位置更新，請在 InfoAdditions 元素中加入下列索引鍵值配對：

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

備註：僅在必要時使用這個功能，因為位置 API 非常耗電。

背景網路

若要在背景執行簡短工作，您的應用程式會將 NativeApplication.nativeApplication.executeInBackground 屬性設定為 true。

例如，您的應用程式啟動檔案上傳作業，然後使用者將其他應用程式移到前面。當應用程式收到上傳完成事件時，就可以將 NativeApplication.nativeApplication.executeInBackground 設定為 false。

將 NativeApplication.nativeApplication.executeInBackground 屬性設定為 true 不保證應用程式將會無限地執行，因為 iOS 會在背景工作上實施時間限制。當 iOS 停止背景處理時，AIR 會傳送 NativeApplication.suspend 事件。

選擇結束背景執行

您的應用程式可以在 InfoAdditions 元素中加入下列索引鍵值配對，藉此明確地選擇結束背景執行：

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

保留的 iOS InfoAdditions 設定

AIR 會在產生的 Info.plist 檔案中設定數個項目，以確保應用程式和執行階段功能正常運作。您無法定義下列設定：

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

備註：您可以定義 MinimumOSVersion。MinimumOSVersion 定義會在 Air 3.3 及更新版本中實行。

支援不同的 iOS 裝置機型

如需 iPad 支援，請在 InfoAdditions 元素內加上 UIDeviceFamily 的適當索引鍵值設定。UIDeviceFamily 設定為字串陣列。每個字串定義支援的裝置。<string>1</string> 設定定義支援 iPhone 與 iPod Touch。<string>2</string> 設定定義支援 iPad。<string>3</string> 設定定義支援 tvOS。如果僅指定其中一個字串，則只會支援該裝置系列。例如，下列設定僅支援 iPad：

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

下列設定同時支援兩種裝置系列 (iPhone/iPod Touch 和 iPad)：

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

此外，在 AIR 3.7 及更新版本中，您可以使用 forceCPURenderModeForDevices 標記，針對指定的裝置組合強制套用 CPU 顯示模式，並針對其餘 iOS 裝置啟用 GPU 顯示模式。

您可以將這個標記新增為 iPhone 標記的子系，並且指定以空格分隔的裝置機型名稱清單。如需有效的裝置機型名稱清單，請參閱 第 191 頁「forceCPURenderModeForDevices」。

例如，若要在舊版 iPod、iPhone 和 iPad 中使用 CPU 模式，並且針對所有其他裝置啟用 GPU 模式，請在應用程式描述器中指定下列項目：

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

高解析度顯示

requestedDisplayResolution 元素指定在具有高解析度螢幕的 iOS 裝置上，應用程式應該使用「標準」還是「高」解析度模式。

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

在高解析度模式下，您可以個別處理高解析度畫面的每一個像素。在標準模式下，應用程式會將裝置螢幕當成標準解析度螢幕。在此模式下繪製單一像素會在高解析度螢幕上設定四個像素的顏色。

預設設定為 standard。請注意，針對以 iOS 為目標的裝置，請將 requestedDisplayResolution 元素當做 iPhone 元素的子元素 (不是 InfoAdditions 元素或 initialWindow 元素) 來使用。

如果您想要在不同裝置使用不同設定，請將您的預設值指定為 requestedDisplayResolution 元素的值。使用 excludeDevices 特質可指定應使用相反值的裝置。例如，透過下列程式碼，高解析度模式可用於所有支援的裝置，除了第 3 代 iPad 以外，因為它使用標準模式：

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

excludeDevices 特質可以在 AIR 3.6 及更新版本中取得。

更多說明主題

第 203 頁「requestedDisplayResolution」

Renaun Erickson：使用 AIR 2.6 開發 iOS 視網膜與非視網膜螢幕 (英文)

iOS 自訂 URI 配置

您可以註冊自訂 URI 配置，以便透過網頁中的連結或裝置上的其他原生應用程式來叫用您的應用程式。若要註冊 URI 配置，請將 CFBundleURLTypes 索引鍵新增至 InfoAdditions 元素。下列範例會註冊名為 com.example.app 的 URI 配置，以使用下列形式的 URL 來叫用應用程式：example://foo。

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

當經由自訂 URI 叫用應用程式時，NativeApplication 物件會傳送 invoke 事件。包括查詢參數的連結 URL，置於 InvokeEvent 物件的 arguments 陣列中。您可以使用任何數量的自訂 URI 配置。

備註：StageWebView 實體中的連結無法開啟使用自訂 URI 配置的 URL。

備註：如果另一個應用程式已註冊某個配置，則您的應用程式無法取代該應用程式成為該 URL 配置的已註冊應用程式。

iOS 相容性篩選

如果您的應用程式只應在具有特定硬體或軟體功能的裝置上使用，請在 InfoAdditions 元素內的 UIRequiredDeviceCapabilities 陣列中新增項目。例如，下列項目指出應用程式需要攝影機和麥克風：

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

如果裝置缺少相對應的功能，則無法安裝應用程式。與 AIR 應用程式相關的功能設定包括：

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

AIR 2.6+ 會自動將 armv7 和 opengles-2 新增至必要功能清單。

備註：這些功能不需要加到應用程式描述器中，就能供應用程式使用。使用 UIRequiredDeviceCapabilities 設定只是為了防止使用者將應用程式安裝到應用程式無法正常運作的裝置上。

結束而非暫停

當使用者離開 AIR 應用程式時，它會進入背景並暫停。如果要讓應用程式完全結束而非暫停，請將 UIApplicationExitsOnSuspend 屬性設為 YES：

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

載入僅限資源的外部 SWF，將下載大小減到最小

AIR 3.7

您可以封裝應用程式所使用的 SWF 子集，並且使用 Loader.load() 方法在執行階段載入其餘 (僅限資源) 的外部 SWF，藉此將初始應用程式下載大小減到最小。若要使用這個功能，您必須封裝應用程式，如此，ADT 就會將所有 ActionScript ByteCode (ABC) 從外部載入的 SWF 檔案移至主應用程式 SWF，留下只包含資源的 SWF 檔案。這是為了符合 Apple Store 禁止在安裝應用程式之後下載任何程式碼的規則所致。

ADT 會執行下列動作以支援外部載入的 SWF (也稱去除的 SWF)：

- 讀取 <iPhone> 元素的 <externalSwfs> 子元素中指定的文字檔，以便存取要在執行時載入、以行分隔的 SWF 清單。

  ```
  <iPhone>
                                                ...

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

- 將每個外部載入的 SWF 中的 ABC 程式碼傳送至主執行檔。
- 忽略 .ipa 檔案中外部載入的 SWF。
- 將去除的 SWF 複製至 .remoteStrippedSWFs 目錄。您可以將這些 SWF 裝載在網站伺服器上，在執行階段，應用程式會在需要時載入 SWF。

透過在文字檔中每行輸入一個名稱的指示方式，指定要在執行階段載入之 SWF 檔案的名稱，如下列範例所示：

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

指定的檔案路徑必須相對於應用程式描述器檔案。此外，您必須在 adt 命令中，將這些 SWF 指定為資源。

備註：這個功能僅適用於標準封裝。為了快速進行封裝 (例如使用解譯器、模擬器或除錯)，ADT 不會建立去除的 SWF。

如需有關這個功能 (包括範例程式碼) 的詳細資訊，請參閱由 Adobe 工程師 Abhinav Dhandh 所張貼的部落格文章，從外部在 iOS 上裝載適用於 AIR 應用程式的次要 SWF。

地理位置支援

如需地理位置支援，請將下列其中一個索引鍵值配對新增至 InfoAdditions 元素：

```
<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>
```

應用程式圖示

下列表格列出每個行動平台所用的圖示大小：

圖示大小	平台
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
114x114	iOS

圖示大小	平台
120x120	iOS
144x144	Android、iOS
152x152	iOS
167x167	iOS
180x180	iOS
192x192	Android
512x512	Android、iOS
1024x1024	iOS

在應用程式描述器檔案的圖示元素中，指定圖示檔案的路徑：

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

如果未提供特定大小的圖示，將使用下一個最大大小的圖示，並縮放為適合的大小。

Android 上的圖示

在 Android 上，應用程式描述器中指定的圖示，將用於應用程式「啟動器」圖示。提供的應用程式「啟動器」圖示應為一組 36x36 像素、48x48 像素、72x72 像素、96x96 像素、144x144 像素和 192x192 像素 PNG 影像。這些圖示依序可用於低解析度、中解析度與高解析度的螢幕。

在 Google Play 商店提交應用程式時，開發人員必須提交 512x512 像素圖示。

iOS 上的圖示

應用程式描述器中定義的圖示，將用於 iOS 應用程式的下列位置：

- 29 x 29 像素圖示：表示低解析度的 Spotlight 搜尋圖示 (iPhone/iPod)，以及表示低解析度的「設定」圖示 (iPad)。
- 40 x 40 像素圖示：表示低解析度的 Spotlight 搜尋圖示 (iPad)。
- 48 x 48 像素圖示：AIR 會在這個影像加上邊框，並用來表示低解析度 iPad 上的 50x50 Spotlight 搜尋圖示。
- 50 x 50 像素圖示：表示低解析度的 Spotlight 搜尋 (iPad)。
- 57 x 57 像素圖示：表示低解析度的應用程式圖示 (iPhone/iPod)。
- 58 x 58 像素圖示：表示 Retina 顯示的 Spotlight 圖示 (iPhone/iPod)，以及表示 Retina 顯示的「設定」圖示 (iPad)。
- 60 x 60 像素圖示：表示低解析度的應用程式圖示 (iPhone/iPod)。
- 72 x 72 像素圖示 (選擇性)：表示低解析度的應用程式圖示 (iPad)。
- 76 x 76 像素圖示 (選擇性)：表示低解析度的應用程式圖示 (iPad)。
- 80 x 80 像素圖示：表示高解析度的 Spotlight 搜尋 (iPhone/iPod/iPad)。
- 100 x 100 像素圖示：表示 Retina 顯示的 Spotlight 搜尋 (iPad)。
- 114 x 114 像素圖示：表示 Retina 顯示的應用程式圖示 (iPhone/iPod)。
- 120 x 120 像素圖示：表示高解析度的應用程式圖示 (iPhone/iPod)。
- 152 x 152 像素圖示：表示高解析度的應用程式圖示 (iPad)。

- 167 x 167 像素圖示：表示高解析度的應用程式圖示 (iPad Pro)。
- 512 x 512 像素圖示：表示低解析度的應用程式圖示 (iPhone/iPod/iPad)。iTunes 會顯示這個圖示。512 像素的 PNG 檔案僅用於測試應用程式的開發版本。當您將最終的應用程式送出至 Apple App Store 時，請以 JPG 檔案個別送出 512 影像。它不會包含在 IPA 之中。
- 1024 x 1024 像素圖示：表示 Retina 顯示的應用程式圖示 (iPhone/iPod/iPad)。

iOS 會對圖示添加閃光效果。您不需要對來源影像套用此效果。若要移除此預設閃光效果，請將以下項目新增至應用程式描述器檔案的 InfoAdditions 元素中：

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

備註：在 iOS 上，應用程式中繼資料會以 png 中繼資料的形式插入應用程式圖示，因此，Adobe 可以追蹤 Apple iOS App Store 中可用的 AIR 應用程式數目。如果您不希望因為此圖示中繼資料，讓應用程式識別為 AIR 應用程式，您必須解除封裝 IPA 檔案，移除圖示中繼資料，然後重新封裝 IPA 檔案。選擇退出 iOS 的 AIR 應用程式分析一文對此程序有較為詳細的說明。

更多說明主題

第 192 頁「icon」

第 193 頁「imageNxN」

Android 開發人員：圖示設計原則

iOS Human Interface Guidelines: Custom Icon and Image Creation Guidelines

iOS 啟動影像

除了應用程式圖示，您還必須提供至少一個命名為 Default.png 的啟動影像。或者，您可以選擇提供不同起始方向、不同解析度 (包括高解析度 Retina 顯示和 16:9 外觀比例) 和不同裝置適用的啟動影像，也可以包含透過 URL 叫用應用程式時，要使用的不同啟動影像。

啟動影像檔不是在應用程式描述器中參考，而是必須放在應用程式根目錄中 (「請勿」將檔案放在子目錄中)。

檔案命名配置
請根據下列配置來命名影像：

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

檔案名稱的 basename 部分是唯一需要的部分，它可能是 Default (大寫 D)，或是您在應用程式描述器的 InfoAdditions 元素中使用 UILaunchImageFile 索引鍵指定的名稱。

「螢幕大小修改選項」部分會指定螢幕的大小 (不是標準螢幕大小之一的時候)。這個修改選項只會套用到具有 16:9 外觀比例的 iPhone 和 iPod touch 機型，例如 iPhone 5 和 iPod touch (第 5 代)。這個修改選項唯一支援的值為 -568h。由於這些裝置支援高解析度 (Retina) 顯示，因此，螢幕大小修改選項一律與也有 @2x 縮放修改選項一起使用。這些裝置的完整預設啟動影像名稱為 Default-568h@2x.png。

urischeme 部分是用來識別 URI 配置的字串。只有在應用程式支援一或多個自訂 URL 配置時才會套用這個部分。例如，如果應用程式可由 example://foo 之類的連結叫用，則使用 -example 做為啟動影像檔案名稱的配置部分。

orientation 部分提供指定多個啟動影像的方式，可根據應用程式啟動時的裝置方向來使用。這個部分只會套用到 iPad 應用程式的影像，可以是下列其中一個值，參考應用程式啟動時裝置的方向：

- -Portrait

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

對於用於高解析度 (Retina) 顯示的啟動影像，scale 部分是 @2x (適用於 iPhone 4、iPhone 5 和 iPhone 6) 或 @3x (適用於 iPhone 6 plus)。(對於用於標準解析度顯示的影像，則完全忽略 scale 部分)。對於如 iPhone 5 和 iPod touch (第 5 代) 較高裝置的啟動影像，您也必須在 basename 部分後面及任何其他部分前面指定螢幕大小修改選項 -528h。

device 部分是用來指定手持裝置和電話的啟動影像。當您的應用程式是同時支援具有單一應用程式二進位資料之手持裝置和數位板的通用應用程式時，會使用這個部分。可能值必須是 ~ipad 或 ~iphone (適用於 iPhone 和 iPod Touch)。

若為 iPhone，您只能包含縱向比例的影像。不過，若是 iPhone 6 plus，還可以新增橫向影像。標準解析度裝置請使用 320x480 像素影像，高解析度裝置則請使用 640x960 像素影像，而 16:9 外觀比例的裝置 (例如 iPhone 5 和 iPod touch (第 5 代))，請使用 640x1136 像素影像。

若是 iPad，您可以包含影像，如下所示：

- AIR 3.3 及更早版本 — 非全螢幕影像：同時包含橫向 (1024x748 表示一般解析度，2048x1496 表示高解析度) 和縱向 (768x1004 表示一般解析度，1536x2008 表示高解析度) 比例的影像。
- AIR 3.4 及更新版本 — 全螢幕影像：同時包含橫向 (1024x768 表示一般解析度，2048x1536 表示高解析度) 和縱向 (768x1024 表示一般解析度，1536x2048 表示高解析度) 比例的影像。請注意，當您封裝非全螢幕應用程式的全螢幕影像時，狀態列會涵蓋最多 20 像素 (最多 40 像素表示高解析度)。請避免在這個區域顯示重要資訊。

範例

下表針對某個可支援最多種裝置與方向，且可透過 example:// 配置的 URL 來啟動的虛構應用程式，顯示您可以包含的一組啟動影像：

檔案名稱	影像大小	用法
Default.png	320 x 480	iPhone，標準解析度
Default@2x.png	640 x 960	iPhone，高解析度
Default-568h@2x.png	640 x 1136	iPhone，高解析度，16:9 外觀比例
Default-Portrait.png	768 x 1004 (AIR 3.3 和更早版本) 768 x 1024 (AIR 3.4 及更新版本)	iPad，縱向
Default-Portrait@2x.png	1536 x 2008 (AIR 3.3 和更早版本) 1536 x 2048 (AIR 3.4 及更新版本)	iPad，高解析度，縱向
Default-PortraitUpsideDown.png	768 x 1004 (AIR 3.3 和更早版本)768 x 1024 (AIR 3.4 及更新版本)	iPad，上下顛倒縱向
Default-PortraitUpsideDown@2x.png	1536 x 2008 (AIR 3.3 和更早版本)1536 x 2048 (AIR 3.4 及更新版本)	iPad，高解析度，上下顛倒縱向
Default-Landscape.png	1024 x 768	iPad，靠左橫向
Default-LandscapeLeft@2x.png	2048 x 1536	iPad，高解析度，靠左橫向

檔案名稱	影像大小	用法
Default-LandscapeRight.png	1024 x 768	iPad，靠右橫向
Default-LandscapeRight@2x.png	2048 x 1536	iPad，高解析度，靠右橫向
Default-example.png	320 x 480	標準 iPhone 上的 example:// URL
Default-example@2x.png	640 x 960	高解析度 iPhone 上的 example:// URL
預設值 - 範例 ~ipad.png	768 x 1004	縱向 iPad 上的 example:// URL
Default-example-Landscape.png	1024 x 768	橫向 iPad 上的 example:// URL

此範例僅說明一種方式。例如，您可將影像 Default.png 用於 iPad，並將 iPhone 與 iPod 的特定啟動影像指定為 Default~iphone.png 與 Default@2x~iphone.png。

請參閱
iOS 應用程式程式設計指南：應用程式啟動影像

要為 **iOS** 裝置封裝的啟動影像

裝置	解析度 (像素)	啟動影像名稱	方向
iPhones			
iPhone 4 (非 Retina)	640 x 960	Default~iphone.png	縱向
iPhone 4、4s	640 x 960	Default@2x~iphone.png	縱向
iPhone 5、5c、5s	640 x 1136	Default-568h@2x~iphone.png	縱向
iPhone6、iPhone7	750 x 1334	Default-375w-667h@2x~iphone.png	縱向
iPhone6+、iPhone7+	1242 x 2208	Default-414w-736h@3x~iphone.png	縱向
iPhone 6+、iPhone7+	2208 x 1242	Default-Landscape-414w-736h@3x~iphone.png	橫向
iPad			
iPad 1、2	768 x 1024	Default-Portrait~ipad.png	縱向
iPad 1、2	768 x 1024	Default-PortraitUpsideDown~ipad.png	上向顛倒縱向
iPad 1、2	1024 x 768	Default-Landscape~ipad.png	靠左橫向
iPad 1、2	1024 x 768	Default-LandscapeRight~ipad.png	靠右橫向
iPad 3、Air	1536 x 2048	Default-Portrait@2x~ipad.png	縱向
iPad 3、Air	1536 x 2048	Default-PortraitUpsideDown@2x~ipad.png	上向顛倒縱向
iPad 3、Air	2048 x 1536	Default-LandscapeLeft@2x~ipad.png	靠左橫向
iPad 3、Air	2048 x 1536	Default-LandscapeRight@2x~ipad.png	靠右橫向
iPad Pro	2048 x 2732	Default-Portrait@2x.png	縱向
iPad Pro	2732 x 2048	Default-Landscape@2x.png	橫向

作品原則
只要尺寸正確，您可以建立任何圖像做為啟動影像。不過，最好還是讓影像與應用程式的起始狀態相符。您甚至可以擷取應用程式啟動畫面的螢幕快照，來建立啟動影像：

1. 在 iOS 裝置上開啟應用程式。顯示第一個使用者介面螢幕時，按住「主畫面」按鈕 (螢幕下方)。按住「主畫面」按鈕的同時，請按住 「電源 / 睡眠」按鈕 (在裝置的最上方)。這樣會擷取一張螢幕快照並傳送至 「相機膠卷」。
2. 藉由從 iPhoto 或其他照片傳送應用程式傳送照片，您可以將影像傳送至您的開發電腦

如果應用程式翻譯成多國語言，請勿在啟動影像中加上任何文字。啟動影像是靜態的，加上的文字可能與其他語言不符。

請參閱
iOS Human Interface Guidelines: Launch images

忽略的設定

行動裝置上的應用程式將忽略適用於原生視窗或桌上型作業系統功能的應用程式設定。忽略的設定如下：

- allowBrowserInvocation
- customUpdateUI
- fileTypes
- height
- installFolder
- maximizable
- maxSize
- minimizable
- minSize
- programMenuFolder
- resizable
- systemChrome
- title
- transparent
- visible
- width
- x
- y

封裝行動 AIR 應用程式

使用 ADT -package 命令，為行動裝置 AIR 應用程式建立應用程式套件。-target 參數可指定建立套件的目標行動平台。

Android 套件
Android 的 AIR 應用程式使用 Android 應用程式套件格式 (APK)，而非 AIR 套件格式。

ADT 使用 APK 目標類型產生的套件，其格式可提交至 Android Market。提交的應用程式必須符合 Android Market 的要求條件，才會被接受。當您建立最終套件之前，應檢視最新的要求條件。請參閱 Android Developers: Publishing on the Market (Android 開發人員：發佈至 Market)。

不同於 iOS 應用程式，您可以使用一般的 AIR 程式碼簽署憑證來簽署 Android 應用程式；不過，若要提交應用程式至 Android Market，憑證必須符合至少有效至 2033 年的 Market 規則。您可以使用 ADT -certificate 命令，建立此類憑證。

若要將應用程式送到不允許應用程式要求使用者從 Google 市場下載 AIR 的替代市場，您可以使用 ADT 的 -airDownloadURL 參數來指定替代的下載 URL。當沒有必要 AIR 執行階段版本的使用者啟動應用程式時，會被導向至指定的 URL。如需詳細資訊，請參閱 第 143 頁 「ADT package 命令」。

根據預設，ADT 會將 Android 應用程式與共用執行階段封裝在一起。因此，若要執行應用程式，使用者應在裝置上安裝個別 AIR 執行階段。

備註：若要強制 ADT 建立使用固定執行階段的 APK，請使用 target apk-captive-runtime。

iOS 套件

iOS 上的 AIR 應用程式使用 iOS 套件格式 (IPA)，而非原生的 AIR 格式。

ADT 使用 ipa-app-store 目標類型與正確程式碼簽署憑證及提供描述檔產生的套件，其格式可提交至 Apple App Store。使用 ipa-ad-hoc 目標類型，封裝臨時發佈的應用程式。

您必須使用正確的 Apple 核發開發人員憑證來簽署您的應用程式。建立測試組建與應用程式提交之前的最終封裝，使用不同的憑證。

如需有關如何使用 Ant 封裝 iOS 應用程式的範例，請參閱 Piotr Walczyszyn：使用 ADT 命令和 ANT 指令碼封裝 iOS 裝置的 AIR 應用程式 (英文)

使用 ADT 封裝

AIR SDK 2.6 和更新版本支援封裝 iOS 和 Android。封裝之前，必須編譯所有的 ActionScript、MXML 與任何擴充程式碼。您也必須具有程式碼簽署憑證。

如需 ADT 命令與選項的詳細參考資訊，請參閱 第 142 頁 「AIR Developer Tool (ADT)」。

Android APK 套件

建立 **APK** 套件

若要建立 APK 套件，請使用 ADT package 命令，設定發行組建的目標類型為 apk，將除錯組建設為 apk-debug，或是將 release-mode 組建設為 apk-emulator 以便在模擬器上執行。

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

在單一行中輸入整個命令；上述範例中的分行僅為便於閱讀之故。此範例也假設 ADT 工具的路徑位於命令列殼層路徑定義中。(如需說明，請參閱 第 263 頁 「Path 環境變數」。)

您必須從包含應用程式檔案的目錄中執行命令。範例中的應用程式檔案為 myApp-app.xml (應用程式描述器檔案)、myApp.swf 與圖示目錄。

如圖所示執行命令時，ADT 將提示您輸入金鑰儲存密碼。(您輸入的密碼字元不會顯示；完成輸入時請按 Enter)。

備註：根據預設，所有 AIR Android 應用程式的套件名稱前置詞均為 air.。若要取消選取這個預設行為，請在您的電腦將環境變數 AIR_NOANDROIDFLAIR 設定為 true。

建立使用原生擴充功能之應用程式的 **APK** 套件

若要為使用原生擴充功能的應用程式建立 APK 套件，除了一般封裝選項以外，還請加入 -extdir 選項。若有多個分享資源 / 元件庫的 ANE，在發出警告之前，ADT 只會選擇單一資源 / 元件庫並忽略其他重複的項目。此選項會指定包含應用程式使用之 ANE 檔案的目錄。例如：

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

建立包含專屬 **AIR** 執行階段版本的 **APK** 套件

若要建立同時包含應用程式與固定 AIR 執行階段版本的 APK 套件，請使用 apk-captive-runtime 目標。此選項會指定包含應用程式使用之 ANE 檔案的目錄。例如：

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

這個技術的潛在缺點包括：

- 當 Adobe 發佈安全性修補程式時，使用者無法自動取得重要的安全性修正
- 佔用更多應用程式 RAM

備註：當您合併執行階段時，ADT 會在應用程式中加入 INTERNET 和 BROADCAST_STICKY 權限。這些權限是 AIR 執行階段所需。

建立除錯 **APK** 套件

若要建立可用於除錯程式的應用程式版本，請使用 apk-debug 作為目標並指定連線選項：

```
adt -package
                                -target apk-debug
                                -connect 192.168.43.45
                                -storetype pkcs12 -keystore ../codesign.p12
                                myApp.apk
                                myApp-app.xml
                                myApp.swf icons
```

-connect 旗標會告知裝置上的 AIR 執行階段，經由網路連線遠端除錯程式的位置。若要經由 USB 除錯，必須指定 -listen 旗標來取代，以指定用於除錯連線的 TCP 連接埠：

```
adt -package
                                -target apk-debug
                                -listen 7936
                                -storetype pkcs12 -keystore ../codesign.p12
                                myApp.apk
                                myApp-app.xml
                                myApp.swf icons
```

為了讓大部分的除錯功能得以運作，也必須在編譯應用程式 SWF 與 SWC 時啟用除錯。如需 -connect 與 -listen 旗標的完整說明，請參閱第 156 頁 「除錯程式連線選項」。

備註：根據預設，ADT 在以 apk-debug 目標封裝應用程式時，會封裝固定 AIR 執行階段與您的 Android 應用程式。若要強制 ADT 建立使用外部執行階段的 APK，請將 AIR_ANDROID_SHARED_RUNTIME 環境變數設定為 true。

在 Android 上，應用程式也必須有存取網際網路的權限，才能經由網路連線執行除錯程式的電腦。請參閱 第 66 頁「Android 權限」。

建立在 **Android** 模擬器上使用的 **APK** 套件

您可以在 Android 模擬器上使用除錯 APK 套件，但無法使用發行模式套件。若要建立在模擬器上使用的發行模式 APK 套件，請使用 ADT package 命令，將目標類型設為 apk-emulator：

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

此範例假設 ADT 工具的路徑位於命令列殼層路徑定義中。(如需說明，請參閱 第 263 頁「Path 環境變數」。)

從 **AIR** 或 **AIRI** 檔案建立 **APK** 套件

您可以從現有 AIR 或 AIRI 檔案建立 APK 套件：

```
adt -target apk -storetype pkcs12 -keystore ../codesign.p12 myApp.apk myApp.air
```

在應用程式描述器檔案中，AIR 檔案必須使用 AIR 2.5 (或更新版本) 命名空間。

建立適用於 **Android x86** 平台的 **APK** 套件

自 AIR 14 開始，可以使用引數 -arch 來封裝適用於 Android x86 平台的 APK。例如：

```
adt     -package
                              -target apk-debug
                              -listen 7936
                              -arch x86
                              -storetype pkcs12 -keystore ../codesign.p12
                              myApp.apk
                              myApp-app.xml
                              myApp.swf icons
```

iOS 套件

在 iOS 上，ADT 會將 SWF 檔案位元組程式碼與其他來源檔案轉換為原生 iOS 應用程式。

1. 使用 Flash Builder、Flash Professional 或命令列編譯器建立 SWF 檔案。
2. 開啟命令殼層或終端機，然後移至 iPhone 應用程式的專案資料夾。
3. 接著，使用 ADT 工具以下列語法建立 IPA 檔案：

```
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
```

變更 adt 參考，使其包含 adt 應用程式的完整路徑。adt 應用程式安裝在 AIR SDK 的 bin 子目錄中。

選取與您想建立之 iPhone 應用程式類型對應的 -target 選項：

- -target ipa-test：選擇此選項可快速編譯應用程式的版本，方便在您的開發人員 iPhone 中進行測試。您也可以使用 ipa-test-interpreter 加快編譯速度，或者使用 ipa-test-interpreter-simulator 在 iOS 模擬器中執行。
- -target ipa-debug：選擇此選項以編譯應用程式的除錯版本，方便在您的開發人員 iPhone 中進行測試。透過此選項，您可以使用除錯工作階段從 iPhone 應用程式接收 trace() 輸出。

您可以包含下列 -connect 選項 (CONNECT_OPTIONS) 之一，指定執行除錯程式的開發電腦 IP 位址：

- -connect：應用程式會嘗試透過 wifi 連線以除錯用於編譯應用程式之開發電腦上的工作階段。
- -connect IP_ADDRESS：應用程式會嘗試透過 wifi 連線以除錯具有指定之 IP 位址電腦上的工作階段。例如：

  ```
  -target ipa-debug -connect 192.0.32.10
  ```

- -connect HOST_NAME：應用程式會嘗試透過 wifi 連線以除錯具有指定之主機名稱電腦上的工作階段。例如：

  ```
  -target ipa-debug -connect bobroberts-mac.example.com
  ```

-connect 為選擇性選項。若未指定，產生的除錯應用程式將不會嘗試連線裝載的除錯程式。或者，您可以指定 -listen (而不是 -connect) 來啟用 USB 除錯，如第 91 頁 「透過 USB 使用 FDB 遠端除錯」中所述。

如果除錯連線嘗試失敗，應用程式將會顯示對話方塊，要求使用者輸入除錯主機的 IP 位址。如果裝置未連線 wifi，連線嘗試將會失敗。如果裝置已連線，但是不在除錯主機的防火牆之後，也會失敗。

您也可以使用 ipa-debug-interpreter 加快編譯速度，或者使用 ipa-debug-interpreter-simulator 在 iOS 模擬器中執行。

如需詳細資訊，請參閱 第 86 頁 「除錯行動 AIR 應用程式」。

- -target ipa-ad-hoc：選擇此選項可建立進行臨時部署的應用程式。請參閱 Apple iPhone 開發人員中心
- -target ipa-app-store：選擇此選項可建立要部署到 Apple App Store 的最終版 IPA 檔。

以應用程式的佈建描述檔路徑來取代 PROFILE_PATH。如需有關佈建描述檔的詳細資訊，請參閱 第 57 頁 「iOS 設定」。

建置時，請使用 -platformsdk 選項指向 iOS 模擬器 SDK，以便在 iOS 模擬器中執行您的應用程式。

取代 SIGNING_OPTIONS 以參照您的 iPhone 開發人員憑證和密碼 請使用下列語法：

```
-storetype pkcs12 -keystore P12_FILE_PATH -storepass PASSWORD
```

以您的 P12 憑證檔案來取代 P12_FILE_PATH。以憑證密碼來取代 PASSWORD。(請參閱以下範例。) 如需 P12 憑證檔案的詳細資訊，請參閱第 167 頁 「將開發人員憑證轉換成 P12 金鑰儲存檔案」。

備註：為 iOS 模擬器封裝時，您可以使用自我簽署的憑證。

取代 APP_DESCRIPTOR 以參照應用程式描述器檔案。

取代 SOURCE_FILES 以參照專案的主要 SWF 檔案，並在後方加上所有要包括的資源。包括您在 Flash Professional 的應用程式設定對話方塊中，或在自訂的應用程式描述器檔案中，所有定義的圖示檔案路徑。同時，請加入起始螢幕圖案檔案 (Default.png)。

使用 -extdir ***extension-directory*** 選項可指定包含應用程式所用 ANE 檔案 (原生擴充功能) 的目錄。如果應用程式未使用原生擴充功能，請勿包含此選項。

重要事項：不要在您的應用程式目錄中建立名為 Resources 的子目錄。執行階段會自動使用這個名稱建立資料夾，以符合 IPA 封裝結構。自行建立 Resources 資料夾將會造成嚴重衝突。

建立用於除錯的 **iOS** 套件

若要建立安裝在測試裝置上的 iOS 套件，請使用 ADT package 命令，將目標類型設為 ios-debug：執行此命令之前，您必須從 Apple 取得開發程式碼簽署憑證與提供描述檔。

```
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
```

備註：您也可以使用 ipa-debug-interpreter 加快編譯速度，或者使用 ipa-debug-interpreter-simulator 在 iOS 模擬器中執行。

在單一行中輸入整個命令；上述範例中的分行僅為便於閱讀之故。此範例也假設 ADT 工具的路徑位於命令列殼層路徑定義中。(如需說明，請參閱 第 263 頁 「Path 環境變數」。)

您必須從包含應用程式檔案的目錄中執行命令。範例中的應用程式檔案為 myApp-app.xml (應用程式描述器檔案)、myApp.swf、圖示目錄與 Default.png 檔案。

您必須使用 Apple 核發的正確發佈憑證來簽署應用程式；不可使用其他程式碼簽署憑證。

使用 -connect 選項可進行 wifi 除錯。應用程式會嘗試透過在指定的 IP 或主機名稱上執行的 Flash 除錯程式 (FDB)，起始除錯工作階段。使用 -listen 選項可進行 USB 除錯。請先啟動應用程式，然後啟動 FDB，如此，就會針對執行中的應用程式起始除錯工作階段。如需詳細資訊，請參閱第 89 頁 「連線至 Flash 除錯程式」。

建立用於 **Apple App Store** 提交的 **iOS** 套件

若要建立提交 Apple App Store 的 iOS 套件，請使用 ADT package 命令，將目標類型設為 ios-app-store：執行此命令之前，您必須從 Apple 取得發佈程式碼簽署憑證與提供描述檔。

```
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
```

在單一行中輸入整個命令；上述範例中的分行僅為便於閱讀之故。此範例也假設 ADT 工具的路徑位於命令列殼層路徑定義中。(如需說明，請參閱 第 263 頁 「Path 環境變數」。)

您必須從包含應用程式檔案的目錄中執行命令。範例中的應用程式檔案為 myApp-app.xml (應用程式描述器檔案)、myApp.swf、圖示目錄與 Default.png 檔案。

您必須使用 Apple 核發的正確發佈憑證來簽署應用程式；不可使用其他程式碼簽署憑證。

重要事項：Apple 要求您使用 Apple Application Loader 程式，以上傳應用程式至 App Store。Apple 僅發佈 Mac OS X 的 Application Loader。因此，使用 Windows 電腦開發 iPhone 的 AIR 應用程式時，您必須使用執行 OS X (版本 10.5.3，或更新的版本) 的電腦，將應用程式提交至 App Store。您可以從 Apple iOS Developer Center 取得 Application Loader 程式。

建立用於臨時發佈的 **iOS** 套件

若要建立臨時發佈的 iOS 套件，請使用 ADT package 命令，將目標類型設為 ios-ad-hoc：執行此命令之前，您必須從 Apple 取得適當的臨時發佈程式碼簽署憑證與提供描述檔。

```
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
```

在單一行中輸入整個命令；上述範例中的分行僅為便於閱讀之故。此範例也假設 ADT 工具的路徑位於命令列殼層路徑定義中。(如需說明，請參閱 第 263 頁 「Path 環境變數」。)

您必須從包含應用程式檔案的目錄中執行命令。範例中的應用程式檔案為 myApp-app.xml (應用程式描述器檔案)、myApp.swf、圖示目錄與 Default.png 檔案。

您必須使用 Apple 核發的正確發佈憑證來簽署應用程式；不可使用其他程式碼簽署憑證。

建立使用原生擴充功能之應用程式的 **iOS** 套件

若要建立使用原生擴充功能之應用程式的 iOS 套件，請搭配使用 ADT package 命令與 -extdir 選項。適當在目標中使用 ADT 命令 (ipa-app-store、ipa-debug、ipa-ad-hoc、ipa-test)。例如：

```
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
```

在單一行中輸入整個命令；上述範例中的分行僅為便於閱讀之故。

至於原生擴充功能，在範例中會假設命名為 extensionsDir 的目錄位於您執行命令的目錄。extensionsDir 目錄包含應用程式使用的 ANE 檔案。

除錯行動 AIR 應用程式

有數種方式可以除錯行動 AIR 應用程式。找出應用程式邏輯問題的最簡單方法，便是使用 ADL 或 iOS 模擬器在開發電腦上進行除錯。您也可以將應用程式安裝在裝置上，使用在桌上型電腦上執行的 Flash 除錯程式進行遠端除錯。

使用 ADL 進行裝置模擬

測試與除錯多數行動應用程式功能的最快、最簡單的方法，便是使用 Adobe Debug Launcher (ADL) 公用程式，在您的開發電腦上執行應用程式。ADL 會使用應用程式描述器中的 supportedProfiles 元素來決定要使用的描述檔。若列出超過一個以上的描述檔，ADL 會使用清單中的第一個。您也可以使用 ADL 的 -profile 參數，選取 supportedProfiles 清單中其中一個其他描述檔。(若未在應用程式描述器中包含 supportedProfiles 元素，則 -profile 引數可指定任何描述檔)。例如，使用下列命令可啟動應用程式，以模擬行動裝置描述檔：

```
adl -profile mobileDevice myApp-app.xml
```

以此種方式在桌上型電腦上模擬行動描述檔時，應用程式執行的環境更符合目標行動裝置。不在行動描述檔中的 ActionScript API 將無法使用。不過，ADL 不會區分不同行動裝置之間的功能。例如，即使您的實際目標裝置不使用軟鍵，仍然可以傳送模擬的 soft-key 按鍵動作至您的應用程式。

ADL 支援裝置方向變更模擬與經由選單命令的軟體輸入模擬。在行動裝置描述檔中執行 ADL 時，ADL 會顯示選單 (在應用程式視窗或桌上型電腦功能列)，讓您輸入裝置旋轉或軟鍵輸入。

軟鍵輸入

ADL 會模擬行動裝置上的「後退」、「選單」及「搜尋」等軟鍵按鈕。當使用行動描述檔啟動 ADL 時，您可以使用顯示的選單，傳送這些按鍵至模擬的裝置。

裝置旋轉

當使用行動描述檔啟動 ADL 時，ADL 可讓您經由顯示的選單，模擬裝置旋轉。您可以向左或向右旋轉模擬的裝置。

旋轉模擬只會影響啟用自動方向的應用程式。您可以在應用程式描述器中，將 autoOrients 元素設定為 true，以啟用此功能。

螢幕大小

您可以設定 ADL -screensize 參數，以便在不同大小的螢幕上測試您的應用程式。您可以傳入其中一種預先定義螢幕類型的程式碼，或傳入代表一般與最大螢幕像素尺寸四個值的字串。

請一律指定縱向版面的像素尺寸，也就是將寬度的值指定為小於高度的值。例如，下列命令將開啟 ADL，模擬 Motorola Droid 上使用的螢幕：

```
adl -screensize 480x816:480x854 myApp-app.xml
```

如需預先定義螢幕類型的清單，請參閱 第 137 頁 「ADL 用法」。

限制

ADL 無法模擬桌上型描述檔上不支援的部分 API。未模擬的 API 包括：

- Accelerometer
- cacheAsBitmapMatrix
- CameraRoll
- CameraUI
- Geolocation
- 在不支援多點觸控及手勢之桌上型作業系統上的多點觸控與手勢
- SystemIdleMode

如果應用程式使用這些類別，應在實際裝置或模擬器上測試功能。

同樣地，有些 API 可以在桌上型電腦的 ADL 下運作，卻無法在所有類型的行動裝置上運作。包括：

- Speex 和 AAC 音效轉碼器
- 輔助功能和螢幕讀取程式支援
- RTMPE
- 載入包含 ActionScript 位元組碼的 SWF 檔案
- PixelBender 著色器

請務必在目標裝置上測試使用這些功能的應用程式，因為 ADL 無法完全複製執行環境。

使用 iOS 模擬器進行裝置模擬

iOS 模擬器 (僅限 Mac) 提供一個快速執行並除錯 iOS 應用程式的方式。當您使用 iOS 模擬器進行測試時，不需要開發人員憑證或佈建描述檔。您仍需要建立 p12 憑證，但該憑證可以自我簽署。

根據預設，ADT 一定會啟動 iPhone 模擬器。若要變更模擬器裝置，請執行下列動作：

- 使用下列命令檢視可用的模擬器。

  ```
  xcrun simctl list devices
  ```

  輸出類似以下所示。

  ```
  == 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)
  ....
  ```

- 您可以如下所示設定環境變數 AIR_IOS_SIMULATOR_DEVICE，以選擇特定模擬器：

```
export AIR_IOS_SIMULATOR_DEVICE = 'iPad Retina'
```

設定環境變數之後，請重新啟動處理程序，並在所選的模擬器裝置上執行應用程式。

備註：在搭配使用 ADT 與 iOS 模擬器時，您必須一律加入 -platformsdk 選項，以指定 iOS 模擬器 SDK 的路徑。

若要在 iOS 模擬器中執行應用程式：

1 搭配使用 adt -package 命令與 -target ipa-test-interpreter-simulator 或 -target ipa-debug-interpreter-simulator，如下列範例所示：

```
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
```

備註：現在，模擬器不再需要簽署選項，因此，可以在 -keystore 旗標中提供任何值，因為 ADT 不會實行。

2 使用 adt -installApp 命令，在 iOS 模擬器中安裝應用程式，如下列範例所示：

```
adt -installApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -package sample_ipa_name.ipa
```

3 使用 adt -launchApp 命令，在 iOS 模擬器中執行應用程式，如下列範例所示：

備註：根據預設，命令 adt -launchApp 會在 iPhone 模擬器中執行應用程式。若要在 iPad 模擬器中執行應用程式，請匯出環境變數 AIR_IOS_SIMULATOR_DEVICE = "iPad"，然後使用命令 adt -launchApp。

```
adt -launchApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -appid sample_ipa_name
```

若要在 iOS 模擬器中測試原生擴充功能，請使用 extension.xml 檔案中的 iPhone-x86 平台名稱並在 nativeLibrary 元素中指定 library.a (靜態元件庫)，如下列 extension.xml 範例所示：

```
<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>
```

備註：在 iOS 模擬器中測試原生擴充功能時，請不要使用針對裝置進行編譯的靜態元件庫 (.a 檔案)。相反地，請務必使用針對模擬器進行編譯的靜態元件庫。

追蹤陳述式

在桌上型電腦上執行您的行動應用程式時，追蹤輸出會列印至除錯程式或用來啟動 ADL 的終端機視窗。在裝置或模擬器上執行您的應用程式時，可以設定遠端除錯工作階段，以檢視追蹤輸出。若有支援，也可使用裝置或作業系統廠商提供的軟體開發工具，以檢視追蹤輸出。

不論何種狀況，編譯應用程式中的 SWF 檔案時必須啟用除錯，讓執行階段可以輸出任何追蹤陳述式。

Android 上的遠端追蹤陳述式

在 Android 裝置或模擬器上執行時，可以使用 Android SDK 包含的 Android Debug Bridge (ADB) 公用程式，檢視 Android 系統記錄中的追蹤陳述式輸出。若要檢視應用程式的輸出，請從開發電腦的命令提示或終端機視窗中執行下列命令：

```
tools/adb logcat air.MyApp:I *:S
```

其中的 MyApp 為應用程式的 AIR 應用程式 ID。引數 *:S 可抑制所有其他處理程序的輸出。除了追蹤輸出之外，若要檢視有關應用程式的系統資訊，可在 logcat 篩選指定中包含 ActivityManager：

```
tools/adb logcat air.MyApp:I ActivityManager:I *:S
```

這些命令範例假設您從 Android SDK 資料夾執行 ADB，或您已新增 SDK 資料夾至 path 環境變數。

備註：在 AIR 2.6+ 中，ADB 公用程式包含在 AIR SDK 中且位於 lib/android/bin 資料夾。

iOS 上的遠端追蹤陳述式

若要從 iOS 裝置上執行的應用程式檢視追蹤陳述式的輸出，必須使用 Flash Debugger (FDB) 建立遠端除錯工作階段。

更多說明主題

Android Debug Bridge：啟用 logcat 記錄

第 263 頁 「Path 環境變數」

連線至 Flash 除錯程式

若要對行動裝置上執行的應用程式進行除錯，可以在您的開發電腦上執行 Flash 除錯程式，並透過網路來連線。若要啟用遠端除錯，必須執行下列項目：

- 在 Android 上，在應用程式描述器中指定 android:permission.INTERNET 權限。
- 啟用除錯來編譯應用程式 SWF。
- 使用 -target apk-debug (適用於 Android) 或 -target ipa-debug (適用於 iOS)，以及 -connect (wifi 除錯) 或 -listen (USB 除錯) 旗標來封裝應用程式。

若是透過 wifi 遠端除錯，裝置必須能夠藉由 IP 位址或完整網域名稱存取執行 Flash 除錯程式的電腦的 TCP 連接埠 7935。若是透過 USB 遠端除錯，裝置必須能夠存取 TCP 連接埠 7936 或 -listen 旗標中指定的連接埠。

對於 iOS，您也可以指定 -target ipa-debug-interpreter 或 -target ipa-debug-interpreter-simulator。

使用 Flash Professional 遠端除錯

應用程式已準備好除錯，且已經在應用程式描述器中設定權限後，請執行下列項目：

1 開啟 「AIR Android 設定」對話方塊。

2 在 「部署」索引標籤下：
   - 為部署類型選取 「除錯」
   - 為 「發佈後」選取 「在連線的 Android 裝置安裝應用程式」
   - 為 「發佈後」取消選取 「在連線的 Android 裝置啟動應用程式」

- 若有需要，設定 Android SDK 的路徑。

3 按一下「發佈」。

您的應用程式將會在裝置上安裝與啟動。

4 關閉「AIR Android 設定」對話方塊。

5 從 Flash Professional 選單，選取「除錯 > 開始遠端除錯工作階段 > ActionScript 3」。

Flash Professional 會在「輸出」面板中顯示「等待 Player 連接」。

6 在裝置上啟動應用程式。

7 在 Adobe AIR 連線對話方塊中，輸入執行 Flash 除錯程式之電腦的 IP 位址或主機名稱，然後按一下「確定」。

透過網路連線使用 FDB 遠端除錯

若要使用命令列 Flash Debugger (FDB) 除錯裝置上執行的應用程式，請先在您的開發電腦上執行除錯程式，然後在裝置上啟動應用程式。以下程序使用 AMXMLC、FDB 與 ADT 工具來編譯、封裝與除錯裝置上的應用程式。範例假設您結合使用 Flex 與 AIR SDK，且 bin 目錄包含在 path 環境變數中。(此假設僅為簡化命令範例)。

1 開啟終端機或命令提示視窗，移至包含應用程式原始碼的目錄。

2 使用 amxmlc 編譯應用程式，並啟用除錯：

```
amxmlc -debug DebugExample.as
```

3 使用 apk-debug 或 ipa-debug 目標來封裝應用程式：

```
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
```

如果永遠使用相同的主機名稱或 IP 位址進行除錯，可以將這些值置於 -connect 旗標之後。應用程式會自動嘗試連線該 IP 位址或主機名稱。否則，每次開始除錯時，必須在裝置上輸入此資訊。

4 安裝應用程式。

在 Android 上，您可以使用 ADT -installApp 命令：

```
adt -installApp -platform android -package DebugExample.apk
```

在 iOS 上，您可以使用 ADT -installApp 命令或 iTunes 來安裝應用程式。

5 在第二個終端機或命令視窗，執行 FDB：

```
fdb
```

6 在 FDB 視窗中，輸入 run 命令：

```
Adobe fdb (Flash Player Debugger) [build 14159]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
                                    Waiting for Player to connect
```

7 在裝置上啟動應用程式。

8 應用程式在裝置或模擬器上啟動後，會開啟 Adobe AIR 連線對話方塊。(封裝應用程式時，若使用 -connect 選項指定主機名稱或 IP 位址，應用程式便會使用該位址嘗試自動連線)。輸入適當的位址並輕點「確定」。

為了在此模式連線除錯程式，裝置必須能夠解析位址或主機名稱，並連線 TCP 連接埠 7935。網路連線為必要。

9 當遠端執行階段連線除錯程式時，可以使用 FDB break 命令來設定中斷點，然後使用 continue 命令開始執行：

```
(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
```

透過 USB 使用 FDB 遠端除錯

AIR 2.6 (Android) AIR 3.3 (iOS)

若要透過 USB 連線除錯應用程式，請使用 -listen 選項替代 -connect 選項來封裝應用程式。若指定 -listen 選項，則當您啟動應用程式時，執行階段會偵聽來自 TCP 連接埠 7936 上的 Flash 除錯程式 (FDB) 連線。然後，請使用 -p 選項執行 FDB，FDB 隨即會起始連線。

適用於 **Android** 的 **USB** 除錯程序

為了讓桌上型電腦上執行的 Flash 除錯程式，可以連線裝置或模擬器上執行的 AIR 執行階段，您必須使用 Android Debug Bridge (ADB，Android SDK 中的公用程式) 或 iOS Debug Bridge (IDB，AIR SDK 中的公用程式)，將裝置連接埠轉送至桌上型電腦連接埠。

1. 開啟終端機或命令提示視窗，移至包含應用程式原始碼的目錄。
2. 使用 amxmlc 編譯應用程式，並啟用除錯：

   ```
   amxmlc -debug DebugExample.as
   ```

3. 使用適當的除錯目標 (例如 apk-debug) 封裝應用程式並指定 -listen 選項：

   ```
   adt -package -target apk-debug -listen -storetype pkcs12 -keystore ../../AndroidCert.p12 DebugExample.apk
   DebugExample-app.xml DebugExample.swf
   ```

4. 透過 USB 纜線將裝置連線至除錯電腦。(您也可以使用此程序來除錯模擬器上執行的應用程式，在此情況下，就不需要也不可能使用 USB)。
5. 安裝應用程式。

   您可以使用 ADT -installApp 命令：

   ```
   adt -installApp -platform android -package DebugExample.apk
   ```

6. 使用 Android ADB 公用程式，從裝置或模擬器轉送 TCP 連接埠 7936 至桌上型電腦：

   ```
   adb forward tcp:7936 tcp:7936
   ```

7. 在裝置上啟動應用程式。
8. 在終端機或命令視窗使用 -p 選項執行 FDB：

   ```
   fdb -p 7936
   ```

9. 在 FDB 視窗中，輸入 run 命令：

   ```
   Adobe fdb (Flash Player Debugger) [build 14159]
                                       Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                       (fdb) run
   ```

10. FDB 公用程式會嘗試連線應用程式。
11. 建立遠端連線時，可以使用 FDB break 命令來設定中斷點，然後使用 continue 命令開始執行：

```
(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
```

備註：AIR 執行階段與 FDB 皆使用連接埠號碼 7936 作為 USB 除錯的預設連接埠。您可以使用 ADT -listen 連接埠參數與 FDB -p 連接埠參數，指定使用不同的連接埠。在此狀況下，您必須使用 Android Debug Bridge 公用程式，將 ADT 中指定的連接埠號碼轉送至 FDB 中指定的連接埠：adb forward tcp:adt_listen_port# tcp:fdb_port#

適用於 **iOS** 的 **USB** 除錯程序

為了讓桌上型電腦上執行的 Flash 除錯程式，可以連線裝置或模擬器上執行的 AIR 執行階段，您必須使用 iOS Debug Bridge (IDB，AIR SDK 中的公用程式)，將裝置連接埠轉送至桌上型電腦連接埠。

1 開啟終端機或命令提示視窗，移至包含應用程式原始碼的目錄。

2 使用 amxmlc 編譯應用程式，並啟用除錯：

```
amxmlc -debug DebugExample.as
```

3 使用適當的除錯目標 (例如 ipa-debug 或 ipa-debug-interpreter) 封裝應用程式並指定 -listen 選項：

```
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 透過 USB 纜線將裝置連線至除錯電腦。(您也可以使用此程序來除錯模擬器上執行的應用程式，在此情況下，就不需要也不可能使用 USB)。

5 在 iOS 裝置上安裝及啟動應用程式。在 AIR 3.4 及更新版本中，您可以使用 adt -installApp 透過 USB 安裝應用程式。

6 使用 idb -devices 命令 (IDB 位於 ***air_sdk_root***/lib/aot/bin/iOSBin/idb) 判斷裝置控制點：

```
./idb -devices
                                    List of attached devices
                                    Handle    UUID
                                        1     91770d8381d12644df91fbcee1c5bbdacb735500
```

備註：(AIR 3.4 及更新版本) 您可以使用 adt -devices (而不是 idb -devices) 來判斷裝置控制點。

7 使用 IDB 公用程式與上一個步驟中發現的裝置 ID，將桌上型電腦上的連接埠轉送至 adt -listen 參數中指定的連接埠 (在此例中，即為 16000，預設為 7936)：

```
idb -forward 7936 16000 1
```

在這個範例中，7936 是桌上型電腦的連接埠，16000 是連線裝置偵聽的連接埠，而 1 是連線裝置的裝置 ID。

8 在終端機或命令視窗使用 -p 選項執行 FDB：

```
fdb -p 7936
```

9 在 FDB 視窗中，輸入 run 命令：

```
Adobe fdb (Flash Player Debugger) [build 23201]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
```

10 FDB 公用程式會嘗試連線應用程式。

11 建立遠端連線時，可以使用 FDB break 命令來設定中斷點，然後使用 continue 命令開始執行：

備註：AIR 執行階段與 FDB 皆使用連接埠號碼 7936 作為 USB 除錯的預設連接埠。您可以使用 IDB -listen 連接埠參數與 FDB -p 連接埠參數，指定使用不同的連接埠。

在行動裝置上安裝 AIR 與 AIR 應用程式

應用程式的一般使用者可以使用裝置的一般應用程式與發佈機制來安裝 AIR 執行階段與 AIR 應用程式。

例如，在 Android 上，使用者可以安裝來自 Android Market 的應用程式。或者，如果使用者在「應用程式」設定中允許安裝不明來源的應用程式，使用者按一下網頁上的連結，或是將應用程式套件複製到裝置上並將它開啟，便能安裝應用程式。如果使用者嘗試安裝 Android 應用程式，但是尚未安裝 AIR 執行階段，便會自動導向 Market，以便從該處安裝執行階段。

在 iOS 上，有兩種方法可以發佈應用程式給一般使用者。主要的發佈通管道是 Apple App Store。您也可以使用臨時發佈，允許限量使用者可以安裝您的應用程式，而毋須經由 App Store。

安裝用於開發的 AIR 執行階段與應用程式

由於行動裝置上的 AIR 應用程式安裝為原生套件，您可以使用一般平台設備來安裝測試用途的應用程式。若有支援，可使用 ADT 命令來安裝 AIR 執行階段與 AIR 應用程式。目前，只在 Android 上支援此做法。

在 iOS 上，您可以使用 iTunes 安裝測試用途的應用程式。測試應用程式必須使用專為應用程式開發所核發的 Apple 程式碼簽屬憑證來簽署，並使用開發提供描述檔來封裝。AIR 應用程式是 iOS 上一個獨立的套件。並不會另外使用分開的執行階段。

使用 **ADT** 安裝 **AIR** 應用程式

開發 AIR 應用程式時，您可以使用 ADT 來安裝與解除安裝執行階段與您的應用程式。(您的 IDE 也可能整合了這些命令，讓您毋須自行執行 ADT)。

您可以使用 AIR ADT 公用程式，在裝置或模擬器上安裝 AIR 執行階段。必須安裝為裝置提供的 SDK。使用 -installRuntime 命令：

```
adt -installRuntime -platform android -device deviceID -package path-to-runtime
```

如果未指定 -package 參數，會從安裝的 AIR SDK 中，選擇適用裝置或模擬器的可用執行階段套件。

若要在 Android 或 iOS 上安裝 AIR 應用程式 (AIR 3.4 及更新版本)，請使用類似的 -installApp 命令：

```
adt -installApp -platform android -device deviceID -package path-to-app
```

為 -platform 引數設定的值應符合您正在安裝的裝置。

備註：重新安裝之前，必須移除 AIR 執行階段或 AIR 應用程式的現有版本。

使用 **iTunes** 在 **iOS** 裝置上安裝 **AIR** 應用程式

若要在 iOS 裝置上安裝 AIR 應用程式進行測試：

1. 開啟 iTunes 應用程式。
2. 如果您尚未這樣做，請將此應用程式的佈建描述檔加到 iTunes 中。在 iTunes 中，選取 「檔案 > 加到資料庫」。然後，選取佈建描述檔 (檔案類型為 mobileprovision)。
3. 在部分 iTunes 版本中，即使您已安裝相同版本的應用程式，新安裝的應用程式並不會取代已安裝的應用程式。在這種情況下，請從裝置及 iTunes 的應用程式清單刪除應用程式。
4. 按兩下應用程式的 IPA 檔。該檔案應該會顯示在 iTunes 的應用程式清單中。
5. 將裝置連接至電腦的 USB 連接埠。
6. 在 iTunes 中，選取裝置的 「應用程式」標籤，並確認在要安裝的應用程式清單中已選取該應用程式。
7. 在 iTunes 應用程式的左側清單選取裝置。然後按一下 「同步」按鈕。完成同步之後，Hello World 應用程式就會顯示在您的 iPhone 中。

如果還是未安裝新的版本，請從裝置及 iTunes 的應用程式清單中刪除該應用程式，然後重做此程序。發生這種情況的原因可能是目前安裝的版本與新安裝的版本使用相同的應用程式 ID 和版本號碼。

更多說明主題

第 151 頁「ADT installRuntime 命令」

第 149 頁「ADT installApp 命令」

在裝置上執行 AIR 應用程式

您可以使用裝置使用者介面，啟動安裝的 AIR 應用程式。若有支援，您也可以使用 AIR ADT 公用程式，從遠端啟動應用程式：

```
adt -launchApp -platform android -device deviceID -appid applicationID
```

-appid 引數的值必須是所要啟動 AIR 應用程式的 AIR 應用程式 ID。請使用 AIR 應用程式描述器中指定的值 (不含封裝時加入的 air. 前置碼)。

如果只附加並執行單一裝置或模擬器，則可以省略 -device 旗標。為 -platform 引數設定的值應符合您正在安裝的裝置。目前，唯一支援值為 android。

移除 AIR 執行階段與應用程式

您可以使用裝置作業系統所提供的一般移除應用程式的方法。若有支援，您也可以使用 AIR ADT 公用程式，移除 AIR 執行階段與應用程式。若要移除執行階段，請使用 -uninstallRuntime 命令：

```
adt -uninstallRuntime -platform android -device deviceID
```

若要解除安裝應用程式，請使用 -uninstallApp 命令：

```
adt -uninstallApp -platform android -device deviceID -appid applicationID
```

如果只附加並執行單一裝置或模擬器，則可以省略 -device 旗標。為 -platform 引數設定的值應符合您正在安裝的裝置。目前，唯一支援值為 android。

設定模擬器

若要在裝置模擬器上執行您的 AIR 應用程式，通常必須使用裝置的 SDK，在您的開發電腦上建立與執行模擬器實體。您之後便可以在模擬器上，安裝 AIR 執行階段的模擬器版本與您的 AIR 應用程式。請注意，模擬器上應用程式的執行速度通常遠慢於實際裝置上的執行速度。

建立 Android 模擬器

1. 啟動 Android SDK 與 AVD Manager 應用程式：
   - 在 Windows 上，執行 Android SDK 根目錄中的 SDK Setup.exe 檔案。
   - 在 Mac OS 上，執行 Android SDK 目錄的 tools 子目錄中的 android 應用程式。
2. 選取「Settings」選項並選取「Force https://」選項。
3. 選取「Available Packages」選項。您應該可以見到可用 Android SDK 的清單。
4. 選取相容的 Android SDK (Android 2.3 或更新的版本) 並按一下「Install Selected」按鈕。
5. 選取「Virtual Devices」選項並按一下「New」按鈕。
6. 進行下列設定：
   - 虛擬裝置的名稱
   - 目標 API，像是 Android 2.3、API level 8
   - SD 卡的大小 (像是 1024)

- 外觀 (像是預設 HVGA)

7 按一下 「Create AVD」按鈕。

請注意，因系統組態而異，「虛擬裝置」的建立可能需要一段時間。

現在，便可以啟動新的 「虛擬裝置」。

1 選取 AVD Manager 應用程式中的 「Virtual Device」。應該會列出上述建立的虛擬裝置。

2 選取 「Virtual Device」並按一下 「Start」按鈕。

3 按一下下一個螢幕上的 「Launch」按鈕。

您應該可以在桌上型電腦上看到一個開啟的模擬器視窗。這可能需要幾秒鐘的時間。起始 Android 作業系統也可能需要一些時間。您可以在模擬器上安裝以 apk-debug 與 apk-emulator 封裝的應用程式。使用 apk 目標封裝的應用程式，無法在模擬器上運作。

更多說明主題

http://developer.android.com/guide/developing/tools/othertools.html#android

http://developer.android.com/guide/developing/tools/emulator.html

更新行動 AIR 應用程式

行動 AIR 應用程式是以原生套件形式散佈，因此會使用平台上其他應用程式的標準更新機制。這通常需要將更新送出到當初用來散佈原始應用程式的同一個市場或 App Store。

行動 AIR 應用程式無法使用 AIR Updater 類別或架構。

更新 Android 上的 AIR 應用程式

針對 Android Market 發佈的應用程式，只要符合下列條件 (這些是 Market 要求的原則，而非 AIR)，將新版本置於 Market 上便能更新應用程式：

- APK 套件是由相同的憑證所簽署。
- AIR ID 相同。
- 應用程式描述器中的 versionNumber 值較大。(若使用 versionLabel 值，也應該讓它遞增)。

從 Android Market 下載應用程式的使用者，其裝置軟體會通知他們有可用的更新。

更多說明主題

Android 開發人員：在 Android Market 上發佈更新

更新 iOS 上的 AIR 應用程式

對於透過 iTunes App Store 散佈的 AIR 應用程式，只要符合下列條件 (這些原則是由 Apple App Store 強制執行，而非 AIR)，便可以送出更新到商店來更新應用程式：

- 程式碼簽署憑證和提供描述檔是核發給同一個 Apple ID
- IPA 套件使用同一個 Apple Bundle ID
- 更新不會減少支援的裝置 (換言之，如果原始應用程式支援執行 iOS 3 的裝置，則您無法建立取消支援 iOS 3 的更新)。

重要事項：由於 AIR SDK 2.6 和更新版本不支援 iOS 3，而 AIR 2 會支援，因此您無法以 AIR 2.6+ 開發的更新來更新以 AIR 2 開發的已發佈 iOS 應用程式。

使用推送通知

推送通知可讓遠端通知提供者將通知傳送到行動裝置上執行的應用程式。AIR 3.4 支援使用 Apple Push Notification service (APNs) 之 iOS 裝置的推送通知。

備註：若要為 AIR for Android 應用程式啟用推送通知，請使用原生擴充功能，例如 as3c2dm，由 Adobe 資訊傳播者 Piotr Walczyszyn 所開發。

本節的其餘部分說明如何在 AIR for iOS 應用程式中啟用推送通知。

備註：此討論內容假設您擁有 Apple 開發人員 ID、熟悉 iOS 開發工作流程，並且曾經在 iOS 裝置上部署至少一個應用程式。

推送通知概觀

Apple Push Notification service (APNs) 可讓遠端通知提供者將通知傳送到 iOS 裝置上執行的應用程式。APNs 支援下列通知類型：

- 警告
- Badges
- 聲音

如需 APNs 的完整資訊，請參閱 developer.apple.com。

在您的應用程式中使用推送通知會牽涉到多個層面：

- 用戶端應用程式 - 註冊推送通知、與遠端通知提供者通訊，以及接收推送通知。
- **iOS** - 管理用戶端應用程式與 APNs 之間的互動。
- **APNs** - 在用戶端註冊期間提供 tokenID，並且將通知從遠端通知提供者傳遞至 iOS。
- 遠端通知提供者 - 將 tokenId 與用戶端的應用程式資訊和推送通知儲存至 APNs。

註冊工作流程

向伺服器端服務註冊推送通知的工作流程如下：

1. 用戶端應用程式要求 iOS 必須啟用推送通知。
2. iOS 將要求轉送至 APNs。
3. APNs 伺服器將 tokenId 傳回至 iOS。
4. iOS 將 tokenId 傳回至用戶端應用程式。
5. 用戶端應用程式 (使用應用程式特定的機制) 會將 tokenId 提供給遠端通知提供者，而遠端通知提供者會儲存 tokenId 以用於推送通知。

通知工作流程

通知工作流程如下：

1. 遠端通知提供者會產生通知，並將通知負荷與 tokenId 一起傳遞至 APNs。
2. APNs 會將通知轉送至裝置上的 iOS。

3 iOS 會將通知負荷推送至應用程式。

推送通知 API

AIR 3.4 引進了一組支援 iOS 推送通知的 API。這些 API 位於 flash.notifications 套件中，並且包含下列類別：

- NotificationStyle - 定義通知類型的常數：ALERT、BADGE 和 SOUND.C
- RemoteNotifier - 可讓您訂閱及取消訂閱推送通知。
- RemoteNotifierSubscribeOptions - 可讓您選取要接收的通知類型。使用 notificationStyles 屬性可定義用來註冊多個通知類型的字串向量。

此外，AIR 3.4 還包括由 RemoteNotifier 所傳送的 flash.events.RemoteNotificationEvent，如下所示：

- 當成功建立應用程式的訂閱及收到來自 APNs 的新 tokenId 時傳送。
- 收到新遠端通知時傳送。

此外，如果在訂閱程序期間發生錯誤，RemoteNotifier 就會傳送 flash.events.StatusEvent。

管理應用程式中的推送通知

若要註冊應用程式以取得推送通知，您必須執行下列步驟：

- 建立可在應用程式中訂閱推送通知的程式碼。
- 啟用應用程式 XML 檔案中的推送通知。
- 建立啟用 iOS Push Services 的佈建描述檔和憑證。

下列註解的範例程式碼會訂閱推送通知並處理推送通知事件：

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

啟用應用程式 XML 檔案中的推送通知

若要在應用程式中使用推送通知，請以 Entitlements 標籤 (在 iphone 標籤底下) 提供下列內容：

```
<iphone>
                                ...
                                   <Entitlements>
                                      <![CDATA[
                                         <key>aps-environment</key>
                                         <string>development</string>
                                      ]]>
                                   </Entitlements>
                                </iphone>
```

當您準備好要將應用程式推送至 App Store 時，請使用 <string> 元素以部署至生產環境：

```
<string>production</string>
```

如果您的應用程式支援當地語系化字串，請在 intialWindow 標籤之下，以 supportedLanguages 標籤指定語言，如下列範例所示：

```
<supportedLanguages>en de cs es fr it ja ko nl pl pt</supportedLanguages>
```

建立啟用 iOS Push Services 的佈建描述檔和憑證

若要啟用應用程式與 APNs 的通訊，您必須以啟用 iOS Push Services 的佈建描述檔和憑證來封裝應用程式，如下所示：

1. 登入您的 Apple 開發人員帳戶。
2. 移至 Provisioning Portal。
3. 按一下 App IDs (應用程式 ID) 索引標籤。
4. 按一下 New App ID (新增應用程式 ID) 按鈕。
5. 指定說明和組合包識別名稱 (組合包識別名稱中不可以使用 *)。
6. 按一下 Submit (送出)。Provisioning Portal 會產生應用程式 ID 並重新顯示應用程式 ID 頁面。
7. 按一下應用程式 ID 右邊的 Configure (憑證)。Configure App ID (設定應用程式 ID) 頁面隨即顯示。
8. 選取 Enable for Apple Push Notification service (啟用 Apple Push Notification 服務) 核取方塊。請注意，您可以使用兩種推送 SSL 憑證：一種可用於開發 / 測試，另一種則適用於生產。
9. 按一下 Development Push SSL Certificate (開發推送 SSL 憑證) 右邊的 Configure (設定) 按鈕。Generate Certificate Signing Request (CSR) (產生憑證簽署要求) 頁面隨即顯示。
10. 使用 Keychain Access 公用程式產生 CSR，如本頁所指示。
11. 產生 SSL 憑證。
12. 下載並安裝 SSL 憑證。
13. (選擇性) 針對生產推送 SSL 憑證重複執行步驟 9 至 12。
14. 按一下 Done (完成)。Configure App ID (設定應用程式 ID) 頁面隨即顯示。
15. 按一下 Done (完成)。App IDs (應用程式 ID) 頁面隨即顯示。請注意您的應用程式 ID 在推送通知內是否出現綠色圓圈。
16. 請務必儲存您的 SSL 憑證，因為稍後應用程式與提供者通訊時會用到。
17. 按一下 Provisioning (佈建) 索引標籤以顯示 Provisioning Profiles (佈建描述檔) 頁面。
18. 為您的新應用程式 ID 建立並下載佈建描述檔。
19. 按一下 Certificates (憑證) 索引標籤並下載新佈建描述檔的新憑證。

使用推送通知的聲音

若要為您的應用程式啟用聲音通知，請合併聲音檔案與想要的任何其他資源，但是以 SWF 和 app-xml 檔案的形式合併在相同的目錄中。例如：

```
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
```

Apple 支援下列聲音資料格式 (aiff、wav 或 caf 檔案)：

- 線性 PCM
- MA4 (IMA/ADPCM)
- uLaw
- aLaw

使用當地語系化的警告通知

若要在應用程式中使用當地語系化的警告通知，請以 lproj 資料夾的形式合併當地語系化字串。例如，您支援西班牙文的警告，如下所示：

1. 在專案內建立與 app-xml 檔案相同層級的 es.lproj 資料夾。
2. 在 es.lproj 資料夾內，建立名為 Localizable.Strings 的文字檔案。
3. 以文字編輯器開啟 Localizable.Strings，然後加入訊息索引鍵和對應的當地語系化字串。例如：

   ```
   "PokeMessageFormat" = "La notificación de alertas en español."
   ```

4. 儲存檔案。
5. 當應用程式收到具有此索引鍵值的警告通知，並且裝置語言為西班牙文時，就會顯示翻譯的警告文字。

設定遠端通知提供者

您需要遠端通知提供者，才能將推送通知傳送至應用程式。這個伺服器應用程式的作用就像提供者一樣，會接受您的推送輸入、將通知和通知資料傳遞至 APNs，然後反過來將推送通知傳送至用戶端應用程式。

如需有關從遠端通知提供者推送通知的詳細資訊，請參閱 Apple Developer Library 中的 使用 Apple Push Notification Service 與提供者通訊。

遠端通知提供者選項

遠端通知提供者的選項如下：

- 根據 APNS-php 開放來源的伺服器建立專屬的提供者。您可以使用 http://code.google.com/p/apns-php/ 設定 PHP 伺服器。這個 Google Code 專案可讓您設計符合特定需求的介面。
- 使用服務提供者。例如，http://urbanairship.com/ 提供現成的 APNs 提供者。註冊這個服務之後，就可以開始使用類似下列的程式碼提供您的裝置字符：

```
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);
                                    }
```

然後，您可以使用 Urban Airship 工具來傳送測試通知。

遠端通知提供者的憑證

您必須將 SSL 憑證和私密金鑰 (先前所產生) 複製到遠端通知提供者伺服器上的適當位置。一般是將這兩個檔案合併到一個 .pem 檔案。若要執行這項作業，請執行下列步驟：

1. 開啟終端機視窗。
2. 輸入下列命令，從 SSL 憑證建立 .pem 檔案：

   ```
   openssl x509 -in aps_developer_identity.cer -inform der -out TestPushDev.pem
   ```

3. 輸入下列命令，建立私密金鑰 (.p12) 的 .pem 檔案：

   ```
   openssl pkcs12 -nocerts -out TestPushPrivateKey.pem -in certificates.p12
   ```

4. 輸入下列命令，將兩個 .pem 檔案合併到一個檔案：

   ```
   cat TestPushDev.pem TestPushPrivateKey.pem > FinalTestPush.pem
   ```

5. 在建立伺服器端推送應用程式時，將合併的 .pem 檔案提供給伺服器提供者。

如需詳細資訊，請參閱 Apple Local and Push Notification Programming Guide 中的在伺服器上安裝 SSL 憑證和金鑰。

處理應用程式中的推送通知

在應用程式中處理推送通知會牽涉到下列層面：

- 全域使用者是否會設定並接受推送通知
- 使用者是否接受個別推送通知
- 推送通知和通知負荷資料的處理方式

設定及接受推送通知

使用者第一次啟動具有推送通知功能的應用程式時，iOS 會顯示含有 Don't Allow (不允許) 和 OK (確定) 按鈕的 **appname Would Like to Send You Push Notifications** (要傳送推送通知給您) 對話方塊。如果使用者選取 OK (確定)，應用程式就會收到已訂閱的所有通知樣式。如果使用者選取 Don't Allow (不允許)，則不會收到任何通知。

備註：此外，使用者也可以前往「設定 > 通知」，控制每個具有推送功能的應用程式可接收的特定通知類型。

Apple 建議，每當應用程式啟用，都應訂閱推送通知。當您的應用程式呼叫 RemoteNotifier.subscribe() 時，會收到 token 類型的 RemoteNotificationEvent，其中包含可唯一識別某個裝置之某個應用程式的 32 位元唯一數字 tokenId。

當裝置收到推送通知時，會顯示「關閉」和「啟動」按鈕的彈出式選單。如果使用者觸碰「關閉」，則不會有任何反應；如果使用者觸碰「啟動」，則 iOS 會叫用應用程式，然後應用程式會收到 notification 類型的 flash.events.RemoteNotificationEvent，如下所示。

推送通知和負荷資料的處理方式

當遠端通知提供者傳送通知到裝置 (使用 tokenID) 時，您的應用程式會收到 notification 類型的 flash.events.RemoteNotificationEvent，不論應用程式是否正在執行。此時，您的應用程式會執行應用程式特定通知處理。如果應用程式會處理通知資料，您就可以透過 JSON 格式的 RemoteNotificationEvent.data 屬性來存取。

第 8 章 開發電視裝置的 AIR 應用程式

電視適用的 AIR 功能

如果電視裝置 (例如電視、數位影像錄影機及藍光播放機) 含有 Adobe AIR for TV，您可以針對裝置建立 Adobe® AIR® 應用程式。AIR for TV 會針對電視裝置最佳化，像是將裝置的硬體加速器應用於高效能視訊與圖形。

電視裝置的 AIR 應用程式為 SWF 類型應用程式，而非 HTML 類型應用程式。您的 AIR for TV 應用程式可以利用硬體加速，以及其他非常適用於 「客廳」環境的 AIR 功能。

裝置描述檔

AIR 使用描述檔來定義類似功能的一組目標裝置。針對 AIR for TV 應用程式使用下列描述檔：

- tv 描述檔。在以 AIR for TV 裝置為目標的 AIR 應用程式中，請使用此描述檔。
- extendedTV 描述檔。如果 AIR for TV 應用程式使用原生擴充功能，請使用此描述檔。

第 211 頁「裝置描述檔」涵蓋了這些描述檔定義的 ActionScript 功能。AIR for TV 的特定 ActionScript 差異詳述於適用於 Adobe Flash Platform 的 ActionScript 3.0 參考中。

如需有關 AIR for TV 描述檔的詳細資訊，請參閱第 121 頁 「支援的描述檔」。

硬體加速

電視裝置會提供硬體加速器，動態提高 AIR 應用程式中的圖形與視訊效能。若要利用這些硬體加速器，請參閱 第 106 頁「AIR for TV 應用程式設計注意事項」。

內容保護

不論是好萊塢賣座佳片，或是獨立製片電影及電視節目，AIR for TV 都可創造出消費者觀賞高品質視訊內容所期待的豐富饗宴。內容提供者可以使用 Adobe 工具建立互動式應用程式。他們可以將 Adobe 伺服器產品整合到內容散佈基礎架構中，或與 Adobe 生態系統的其中一家合作夥伴合作。

內容保護是散佈高品質視訊的重要條件。AIR for TV 支援 Adobe® Flash® Access™，這是一種內容保護和付費解決方案，可滿足內容擁有者 (包括主要電影製片公司) 嚴格的安全需求。

Flash Access 支援下列功能：

- 視訊串流與下載。
- 各種商業模式，包括廣告贊助、訂閱、租借及電子銷售。
- 不同的內容傳遞技術，包括 HTTP 動態串流、使用 Flash® Media Server 透過 RTMP (即時媒體通訊協定) 的串流，以及使用 HTTP 的漸進式下載。

針對現有較低安全需求的串流解決方案，AIR for TV 也內建 RTMPE (加密版的 RTMP) 支援。Flash Media Server 中支援 RTMPE 及相關的 SWF 驗證技術。

如需詳細資訊，請參閱 Adobe Flash Access。

多頻道音效

從 AIR 3 開始，AIR for TV 支援依進度從 HTTP 伺服器下載的視訊的多頻道音效。這項支援包括下列轉碼器：

- AC-3 (杜比數位)
- E-AC-3 (加強杜比音效)
- DTS Digital Surround
- DTS Express
- DTS-HD High Resolution Audio
- DTS-HD Master Audio

備註：尚未在從 Adobe Flash Media Server 串流的視訊中提供多頻道音效的支援。

遊戲輸入

從 AIR 3 開始，AIR for TV 支援 ActionScript API，讓應用程式可以透過附加的遊戲輸入裝置 (例如搖桿、遊戲台和遊戲棒) 進行通訊。雖然這些裝置都稱為遊戲輸入裝置，但是任何 AIR for TV 應用程式 (而非只有遊戲) 都可以使用裝置。

各種遊戲輸入裝置提供不同的功能。因此，裝置會在 API 中進行一般化，以便應用程式可以搭配不同 (及可能未知) 的遊戲輸入裝置正常地運作。

GameInput 類別是遊戲輸入 ActionScript API 的輸入點。如需詳細資訊，請參閱 GameInput。

Stage 3D 加速圖形顯示方式

從 AIR 3 開始，AIR for TV 支援 Stage 3D 加速圖形顯示方式。Stage3D ActionScript API 是一組啟用進階 2D 和 3D 功能的低階 GPU 加速 API。這些低階 API 讓開發人員可以彈性地利用 GPU 硬體加速，顯著改善效能。此外，您也可以使用支援 Stage3D ActionScript API 的遊戲引擎。

如需詳細資訊，請參閱 Gaming engines, 3D, and Stage 3D (遊戲引擎、3D 和 Stage 3D)。

原生擴充功能

應用程式以 extendedTV 描述檔為目標時，可以使用 ANE (AIR 原生擴充功能) 套件。

通常，裝置製造商會提供 ANE 套件，以供存取 AIR 不支援的裝置功能。例如，原生擴充功能可讓您切換電視頻道，或是暫停播放視訊播放程式。

當您封裝使用 ANE 套件的 AIR for TV 應用程式時，應用程式會封裝到 AIRN 檔，而非 AIR 檔。

AIR for TV 裝置的原生擴充功能永遠是 「組合裝置」的原生擴充功能。組合裝置表示擴充功能元件庫是安裝在 AIR for TV 裝置上。包含在應用程式套件中的 ANE 套件 「絕不會」包含擴充功能的原生元件庫。有時候，則會包含僅有 ActionScript 的原生擴充功能版本。僅有 ActionScript 版本是虛設常式或擴充功能的模擬。裝置製造商會在裝置上安裝實際的擴充功能，包括原生元件庫。

如果您要開發原生擴充功能，請注意下列事項：

- 如果要建立裝置適用的 AIR for TV 原生擴充功能，永遠洽詢裝置製造商。
- 在某些 AIR for TV 裝置上，只有裝置製造商能建立原生擴充功能。
- 在所有 AIR for TV 裝置上，都由裝置製造商決定要安裝哪些原生擴充功能。
- 建置 AIR for TV 原生擴充功能的開發工具會依製造商而有所不同。

如需有關在 AIR 應用程式中使用原生擴充功能的詳細資訊，請參閱第 128 頁 「使用 Adobe AIR 的原生擴充功能」。

如需有關建立原生擴充功能的詳細資訊，請參閱 Developing Native Extensions for Adobe AIR。

AIR for TV 應用程式設計注意事項

視訊注意事項

視訊編碼原則

將視訊串流至電視裝置時，Adobe 建議採取下列編碼原則：

視訊轉碼器：	H.264，主要或高描述檔，漸進式編碼
解析度：	720i、720p、1080i 或 1080p
影格速率：	每秒 24 格，或每秒 30 格
音效轉碼器：	AAC-LC 或 AC-3、44.1 kHz、立體聲或這些多頻道音效解碼器：E-AC-3、DTS、DTS Express、DTS-HD High Resolution Audio 或 DTS-HD Master Audio
混合位元速率：	視可用頻寬而定，最高 8M bps
音效位元速度：	最高 192 Kbps
像素外觀比例：	1 × 1

針對傳遞至 AIR for TV 裝置的視訊，Adobe 建議採用 H.264 轉碼器。

備註：AIR for TV 也支援以 Sorenson Spark 或 On2 VP6 轉碼器編碼的視訊。不過，硬體無法解碼和顯示這些轉碼器。相反地，執行階段會使用軟體來解碼並顯示這些轉碼器，因此，視訊會以極慢的影格速率播放。因此，請儘可能使用 H.264。

StageVideo 類別

AIR for TV 支援直接在硬體中解碼和顯示 H.264 編碼視訊。您可以使用 StageVideo 類別來啟用此功能。

請參閱 「ActionScript 3.0 開發人員指南」的 使用硬體加速呈現方式的 StageVideo 類別，以取得下列方面的詳細資訊：

- StageVideo 類別及相關類別的 API。
- StageVideo 類別的使用限制。

為了讓使用 H.264 編碼視訊 Video 物件的現有 AIR 應用程式獲得最佳支援，AIR for TV 會 「在內部」使用 StageVideo 物件。此作法可讓視訊播放從硬體解碼和顯示中獲益。不過，Video 物件有著與 StageVideo 物件相同的限制。例如，如果應用程式嘗試旋轉視訊，視訊並不會旋轉，因為視訊是由硬體顯示，而非執行階段。

不過，當您撰寫新的應用程式時，請針對 H.264 編碼視訊使用 StageVideo 物件。

如需 StageVideo 類別的使用範例，請參閱在電視上傳遞 Flash Platform 的視訊與內容。

視訊傳遞原則

在 AIR for TV 裝置上播放視訊期間，網路的可用頻寬可能會有所變化。例如，可能有其他使用者開始使用同一條網際網路連線，而使這類變化產生。

因此，Adobe 建議讓視訊傳遞系統使用最適化位元傳輸速率功能。例如，在伺服端上，Flash Media Server 支援最適化位元傳輸速率功能。在用戶端，您可以使用 Open Source Media Framework (OSMF)。

以下是可用來透過網路，將視訊內容傳遞至 AIR for TV 應用程式的通訊協定：

- HTTP 和 HTTPS 動態串流 (F4F 格式)
- RTMP、RTMPE、RTMFP、RTMPT 和 RTMPTE Streaming
- HTTP 和 HTTPS 漸進式下載

如需詳細資訊，請參閱下列章節：

- Adobe Flash Media Server 開發人員指南
- Open Source Media Framework

音效注意事項

在 AIR for TV 應用程式中，用於播放聲音的 ActionScript 與在其他 AIR 應用程式中使用的相同。如需詳細資訊，請參閱「ActionScript 3.0 開發人員指南」的處理聲音。

關於 AIR for TV 中的多頻道音效支援，請注意下列幾點：

- AIR for TV 支援依進度從 HTTP 伺服器下載的視訊的多頻道音效。尚未在從 Adobe Flash Media Server 串流的視訊中提供多頻道音效的支援。
- 雖然 AIR for TV 支援許多音效解碼器，但並非所有 AIR for TV 裝置都支援整個組合。使用 flash.system.Capabilities 方式 hasMultiChannelAudio() 來檢查 AIR for TV 裝置是否支援特定多管道音效解碼器，例如 AC-3。

  例如，考量會依進度從伺服器下載視訊檔案的應用程式。伺服器有不同的 H.264 視訊檔，可支援各種多頻道的音效解碼器。應用程式會使用 hasMultiChannelAudio() 來判斷要向伺服器幽球哪一個視訊檔案。或者，應用程式可將 Capabilities.serverString 中包含的字串傳送給伺服器。字串表示是否還有多頻道音效解碼器，讓伺服器選擇適當的視訊檔案。
- 使用其中一個 DTS 音效轉碼器時，會出現 hasMultiChannelAudio() 傳回 true 但無法播放 DTS 音效的情形。

  例如，請考慮具有 S/PDIF 輸出的 Blu-ray 播放程式連接到舊版擴音器。舊版擴音器不支援 DTS，但是 S/PDIF 沒有通訊協定可通知 Blu-ray 播放程式。如果 Blu-ray 播放程式傳送 DTS 串流到舊版擴音器，使用者不會聽到任何聲音。因此，使用 DTS 時，最好的方法就是提供使用者介面，讓使用者可以指示是否播放出任何聲音。然後，您的應用程式就能回復至不同的轉碼器。

下表摘述在 AIR for TV 應用程式中，使用不同音效轉碼器的時機。表格也表示 AIR for TV 裝置使用硬體加速器來解碼視訊解碼器的時間。硬體解碼可改善效能，減輕 CPU 負載。

音效轉碼器	在 AIR for TV 裝置上的可用性	硬體解碼	何時該使用此音效轉碼器	詳細資訊
AAC	永遠	永遠	在以 H.264 編碼的視訊中。 音效串流，例如網際網路音樂串流服務。	使用純音效 AAC 串流時，請將音效串流封裝在 MP4 容器中。
mp3	永遠	否	應用程式 SWF 檔中的聲音。 在以 Sorenson Spark 或 On2 VP6 編碼的視訊中。	使用 mp3 音效的 H.264 視訊無法在 AIR for TV 裝置上播放。

音效轉碼器	在 AIR for TV 裝置上的可用性	硬體解碼	何時該使用此音效轉碼器	詳細資訊
AC-3 (杜比數位) E-AC-3 (加強杜比音效) DTS Digital Surround DTS Express DTS-HD High Resolution Audio DTS-HD Master Audio	檢查	是	在以 H.264 編碼的視訊中。	通常，AIR for TV 會將多管道視訊串流傳到解碼與播放音效的外部音效 / 視訊接收器。
Speex	永遠	否	接收即時語音串流。	使用 Speex 音效的 H.264 視訊無法在 AIR for TV 裝置上播放。請僅與 Sorenson Spark 或 On2 VP6 編碼視訊搭配使用 Speex。
NellyMoser	永遠	否	接收即時語音串流。	使用 NellyMoser 音效的 H.264 視訊無法在 AIR for TV 裝置上播放。請僅與 Sorenson Spark 或 On2 VP6 編碼視訊搭配使用 NellyMoser。

備註：某些視訊檔案包含兩個音效串流。例如，視訊檔可同時包含 AAC 串流與 AC3 串流。AIR for TV 並不支援這樣的視訊檔案，且使用這類檔案可能會導致視訊沒有聲音。

圖形硬體加速

使用硬體圖形加速

AIR for TV 裝置提供 2D 圖形作業的硬體加速功能。本裝置的硬體圖形加速器會執行下列作業，以減輕 CPU 的負擔：

- 點陣圖顯示
- 點陣圖縮放
- 點陣圖混合
- 矩形實心填滿

此硬體圖形加速可讓 AIR for TV 應用程式中的許多圖形作業展現高度效能。其中部分作業包括：

- 滑動轉場
- 縮放轉場
- 淡入與淡出
- 以 Alpha 複合多個影像

若要運用硬體圖形加速來提高這幾種作業的效能，請採取下列其中一種技術：

- 在 MovieClip 物件和其他內容大多不變的顯示物件上，將 cacheAsBitmap 屬性設為 true。然後，在這些物件上執行滑動轉場、漸淡轉場和 Alpha 混合。
- 在您要縮放或轉譯 (套用 x 和 y 重新定位) 的顯示物件上，使用 cacheAsBitmapMatrix 屬性。

  裝置的硬體加速器會藉由使用 Matrix 類別作業進行縮放和轉譯，來執行這些作業。再設想另一種情況，假設顯示物件的 cacheAsBitmap 屬性設為 true，而您變更該物件的尺寸。當尺寸變更時，執行階段軟體會重繪點陣圖。使用軟體重繪的效能，會低於使用 Matrix 作業進行硬體加速縮放的效能。

例如，假設應用程式顯示的影像會在使用者選取影像時放大。多使用 Matrix 縮放作業幾次，即可得到影像正在放大的效果。不過，視原始影像和最終影像的大小而定，最終影像的品質可能不理想。因此，完成放大作業後，請重設顯示物件的尺寸。因為 cacheAsBitmap 為 true，所以執行階段軟體會重繪顯示物件，但僅重繪一次，而且會顯示高品質影像。

備註：通常，AIR for TV 裝置並不支援硬體加速的旋轉和傾斜。因此，如果您在 Matrix 類別中指定旋轉和傾斜，AIR for TV 就會在軟體中執行所有的 Matrix 作業。這些軟體作業會對效能產生不利的影響。

- 使用 BitmapData 類別以建立自訂點陣圖快取行為。

如需有關點陣圖快取的詳細資訊，請參閱下列章節：

- 快取顯示物件
- 點陣圖快取
- 手動點陣圖快取

管理圖形記憶體

為了執行圖形加速作業，硬體加速器會使用特殊的圖形記憶體。如果應用程式用盡所有圖形記憶體，則應用程式執行速度會變慢，因為 AIR for TV 會退一步使用軟體來處理圖形作業。

若要管理應用程式的圖形記憶體使用情形，您可以：

- 在使用完影像或其他點陣圖資料後，釋放相關的圖形記憶體。若要執行這項操作，請呼叫 Bitmap 物件之 bitmapData 屬性的 dispose() 方法。例如：

  ```
  myBitmap.bitmapData.dispose();
  ```

  備註：釋放 BitmapData 物件的參考並不會立即釋放圖形記憶體。執行階段的記憶體回收程式最後還是會釋放圖形記憶體，但是呼叫 dispose() 讓應用程式有更大的控制權。

- 使用 Adobe 提供的 AIR 應用程式 PerfMaster Deluxe，充分瞭解目標裝置上的硬體圖形加速。此應用程式會顯示各種作業執行時的每秒影格數。請使用 PerfMaster Deluxe 來比較相同作業的不同實作。例如，比較移動點陣圖影像與移動向量影像有何不同。您可以從電視版 Flash Platform 取得 PerfMaster Deluxe。

管理顯示清單

若要讓顯示物件變成不可見的，請將物件的 visible 屬性設定為 false。然後，物件仍會在顯示清單上，但 AIR for TV 不會呈現或顯示它。這項技術適用於經常往返於檢視的物件，因為它只會產生些微處理負荷。不過，將 visible 屬性設為 false 並不會釋放物件的任何資源。因此，當您完成顯示某個物件，或者至少有一段時間不再需要它，請從顯示清單移除該物件。另外，請將物件的所有參照設定為 null。這些動作可讓記憶體回收程式釋放物件的資源。

PNG 與 JPEG 影像用法

PNG 和 JPEG 是應用程式中常見的兩種影像格式。關於 AIR for TV 應用程式中的這些影像格式，請注意下列幾點：

- AIR for TV 通常使用硬體加速來解碼 JPEG 檔。
- AIR for TV 通常使用軟體來解碼 PNG 檔。在軟體中解碼 PNG 檔的速度很快。
- PNG 是唯一支援透明度 (Alpha 色版) 的跨平台點陣圖格式，

因此，請在應用程式中依下列原則使用這些影像格式：

- 使用相片的 JPEG 檔可從硬體加速解碼中獲益。
- 將 PNG 影像檔用於使用者介面元素。使用者介面元素可能有 Alpha 設定，而軟體解碼可以提供足夠的效能來顯示使用者介面元素。

AIR for TV 應用程式的舞台

編寫 AIR for TV 應用程式時，請在使用 Stage 類別時注意下列幾點：

- 螢幕解析度
- 安全檢視區域
- 舞台縮放模式
- 舞台對齊方式
- 舞台顯示狀態
- 針對多種螢幕大小進行設計
- 舞台品質設定

螢幕解析度

目前電視裝置通常有這幾種螢幕解析度：540p、720p 及 1080p。這些螢幕解析度會在 ActionScript Capabilities 類別中產生下列值：

螢幕解析度	Capabilities.screenResolutionX	Capabilities.screenResolutionY
540p	960	540
720p	1280	720
1080p	1920	1080

若要針對特定裝置撰寫全螢幕的 AIR for TV 應用程式，請將 Stage.stageWidth 和 Stage.stageHeight 明確設為裝置的螢幕解析度。不過，若要撰寫在多個裝置上執行的全螢幕應用程式，請使用 Capabilities.screenResolutionX 和 Capabilities.screenResolutionY 屬性來設定舞台尺寸。

例如：

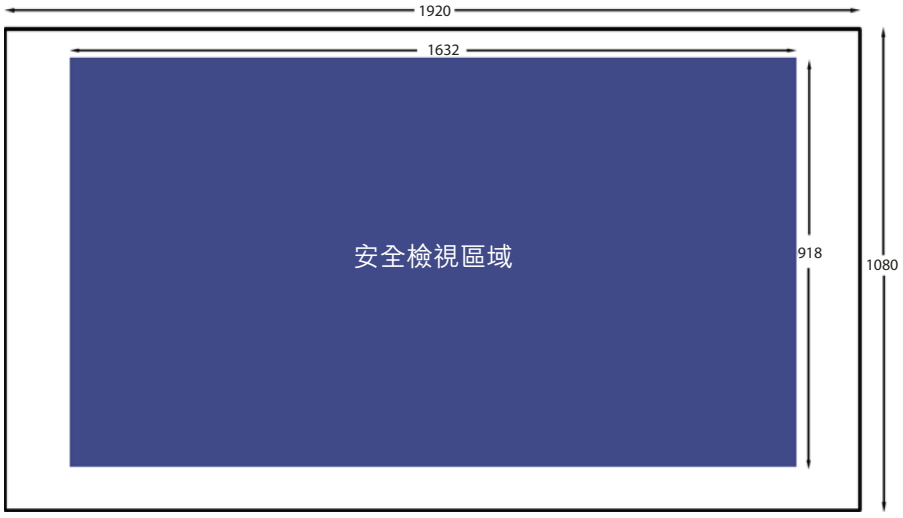
```
stage.stageWidth = Capabilities.screenResolutionX;
stage.stageHeight = Capabilities.screenResolutionY;
```

安全檢視區域

電視上的安全檢視區域是指離螢幕邊緣有段內移距離的螢幕區域。此區域的內移距夠遠，讓使用者可以看見整個區域，而不會被電視邊框遮住任何部分。因為各家製造商的邊框 (螢幕周圍的實體外框) 不相同，必要的內移距也會不同。安全檢視區域的存在，是要確保有一塊絕對看得見的螢幕區域。安全檢視區域又稱為「標題安全區域」。

超出掃描區域是指被邊框遮住而看不見的螢幕區域。

Adobe 建議螢幕四邊的內移距都是 7.5%。例如：



螢幕解析度 1920 x 1080 的安全檢視區域

設計全螢幕的 AIR for TV 應用程式時，請務必考量安全檢視區域：

- 以整個螢幕為背景，例如背景影像或背景顏色。
- 只在安全檢視區域放置重要的應用程式元素，例如文字、圖形、視訊及使用者介面項目 (如按鈕)。

下表顯示各常見螢幕解析度的安全檢視區域尺寸 (使用內移距 7.5%)。

螢幕解析度	安全檢視區域的寬度和高度	左右方內移距寬度	上下方內移距高度
960 x 540	816 x 460	72	40
1280 x 720	1088 x 612	96	54
1920 x 1080	1632 x 918	144	81

不過，最佳作法是一律動態計算安全檢視區域。例如：

```
var horizontalInset, verticalInset, safeAreaWidth, safeAreaHeight:int;

horizontalInset = .075 * Capabilities.screenResolutionX;
verticalInset = .075 * Capabilities.screenResolutionY;
safeAreaWidth = Capabilities.screenResolutionX - (2 * horizontalInset);
safeAreaHeight = Capabilities.screenResolutionY - (2 * verticalInset);
```

舞台縮放模式

將 Stage.scaleMod 設為 StageScaleMode.NO_SCALE，然後偵聽舞台調整大小事件。

```
stage.scaleMode = StageScaleMode.NO_SCALE;
stage.addEventListener(Event.RESIZE, layoutHandler);
```

此設定會使得舞台座標就是像素座標。搭配 FULL_SCREEN_INTERACTIVE 顯示狀態和 TOP_LEFT 舞台對齊方式，此設定可讓您有效地使用安全檢視區域。

具體而言，在全螢幕應用程式中，此縮放模式表示 Stage 類別的 stageWidth 和 stageHeight 屬性會對應於 Capabilities 類別的 screenResolutionX 和 screenResolutionY 屬性。

此外，當應用程式的視窗改變大小時，舞台內容仍會保持原本定義的大小。執行階段不會自動進行版面或縮放調整。另外，當視窗改變大小時，執行階段會傳送 Stage 類別的 resize 事件。因此，當應用程式開始時和應用程式視窗調整大小時，您完全可以控制調整應用程式內容的方式。

備註：NO_SCALE 行為在任何 AIR 應用程式中都相同。不過，在 AIR for TV 應用程式中，使用此設定對於使用安全檢視區域而言十分重要。

舞台對齊方式

將 Stage.align 設為 StageAlign.TOP_LEFT：

```
stage.align = StageAlign.TOP_LEFT;
```

此對齊方式將螢幕左上角定為 0,0 座標，以便使用 ActionScript 來放置內容。

搭配 NO_SCALE 縮放模式和 FULL_SCREEN_INTERACTIVE 顯示狀態，此設定可讓您有效地使用安全檢視區域。

舞台顯示狀態

將全螢幕 AIR for TV 應用程式中的 Stage.displayState 設為 StageDisplayState.FULL_SCREEN_INTERACTIVE：

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

此值會設定 AIR 應用程式將舞台展開佈滿整個螢幕，並允許使用者輸入。

Adobe 建議使用 FULL_SCREEN_INTERACTIVE 設定。搭配 NO_SCALE 縮放模式和 TOP_LEFT 舞台對齊方式，此設定可讓您有效地使用安全檢視區域。

因此，對於全螢幕應用程式，請在主要文件類別上 ADDED_TO_STAGE 事件的處理常式中執行下列動作：

```
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;
}
```

然後，在 RESIZE 事件的處理常式中：

- 比較螢幕解析度大小與舞台的寬度和高度。如果相同，則表示 RESIZE 事件發生的原因是舞台顯示狀態已變更為 FULL_SCREEN_INTERACTIVE。
- 計算並儲存安全檢視區域及對應的內移距尺寸。

```
private function onResize(evt:Event):void
{
    if ((Capabilities.screenResolutionX == stage.stageWidth) &&
        (Capabilities.screenResolutionY == stage.stageHeight))
    {

        // Calculate and save safe viewing area dimensions.
    }
}
```

當舞台尺寸等於 Capabilities.screenResolutionX 和 screenResolutionY 時，AIR for TV 會使硬體為您的視訊和圖形傳遞最佳的真實度。

備註：Capabilities.screenResolutionX 和 screenResolutionY 值而有所不同，那兩個值取決於執行 AIR for TV 的裝置。例如，執行 AIR for TV 的機上盒螢幕解析度為 1280 x 720，而第四台螢幕解析度為 1920 x 1080。不過，AIR for TV 會使硬體傳遞最佳真實度。因此，在這個範例中，使用 1920 x 1080 螢幕解析度的硬體會顯示 1080p 視訊。

針對多種螢幕大小進行設計

您可以開發在多個 AIR for TV 裝置上都能正常執行和精美呈現的全螢幕 AIR for TV 應用程式。請執行下列動作：

1. 將舞台屬性 scaleMode、align 及 displayState 設為建議值：StageScaleMode.NO_SCALE、StageAlign.TOP_LEFT 及 StageDisplayState.FULL_SCREEN_INTERACTIVE。
2. 根據 Capabilities.screenResolutionX 和 Capabilities.screenResolutionY 來設定安全檢視區域。
3. 根據安全檢視區域的寬度和高度，調整內容的大小和版面。

   雖然內容的物件會很大 (尤其是與行動裝置應用程式比較)，但動態版面、相對位置及最適化內容等的概念都相同。如需 ActionScript 支援這些概念的進一步資訊，請參閱編寫多種螢幕尺寸適用的行動 Flash 內容。

舞台品質

AIR for TV 應用程式的 Stage.quality 屬性永遠是 StageQuality.High。您無法改變。

此屬性指定所有 Stage 物件的顯示品質。

遙控器輸入處理

使用者通常以遙控器來與 AIR for TV 應用程式互動。不過，處理按鍵輸入的方式，與處理桌上型應用程式中鍵盤按鍵輸入的方式相同。具體而言，請處理事件 KeyboardEvent.KEY_DOWN。如需詳細資訊，請參閱 「ActionScript 3.0 開發人員指南」的擷取鍵盤輸入。

遙控器的按鍵對應於 ActionScript 常數。例如，遙控器方向鍵台的按鍵對應如下：

遙控器的方向鍵台按鍵	ActionScript 3.0 常數
向上	Keyboard.UP
向下	Keyboard.DOWN
靠左	Keyboard.LEFT
靠右	Keyboard.RIGHT
「確定」或「選取」	Keyboard.ENTER

AIR 2.5 已新增其他許多 Keyboard 常數來支援遙控器輸入。如需完整清單，請參閱 「適用於 Adobe Flash Platform 的 ActionScript 3.0 參考」的 Keyboard 類別。

為了儘可能確保應用程式可以在許多裝置上執行，Adobe 建議下列作法：

- 儘可能只使用方向鍵台按鍵。

  不同的遙控器裝置會有一組不同的按鍵。不過，這些遙控器裝置通常一定有方向鍵台按鍵。

  例如，藍光播放機的遙控器通常沒有 「頻道上移」和 「頻道下移」鍵。此外，並不是所有遙控器都有播放、暫停及停止鍵。

- 如果方向鍵台按鍵不敷應用程式所需，請使用 「選單」和 「資訊」鍵。

  「選單」和 「資訊」鍵是第二常見的遙控器按鍵。

- 考量使用萬用遙控器的頻率。

  即使是針對特定裝置建立應用程式，也必須瞭解許多使用者並不使用裝置隨附的遙控器。他們使用的是萬用遙控器。而且，使用者在萬用遙控器上為各按鍵設定的功能，不一定與裝置遙控器的按鍵功能相同。因此，建議只使用最常見的按鍵。

- 確定使用者永遠可以使用其中一個方向鍵台按鍵來跳離情境。

  有時候，應用程式不得不使用常見遙控器按鍵以外的其他按鍵。使用其中一個方向鍵台按鍵作為跳離途徑，可確保應用程式在所有裝置上都不會有使用卡住的問題。

- 不需要指標輸入，除非您知道目標 AIR for TV 具有指標輸入功能。

  雖然許多桌上型應用程式會需要滑鼠輸入，但是大多數的電視都不支援指標輸入。因此，如果您要將桌上型應用程式轉換為在電視上執行，請務必將應用程式修改為不需要滑鼠輸入。這些修改包括變更事件處理方式和對使用者顯示的指示。例如，當應用程式的啟動畫面出現時，請不要顯示「按一下以開始」這類文字。

管理焦點

當桌上型應用程式中的使用者介面元素取得焦點時，就成為使用者輸入事件 (如鍵盤和滑鼠事件) 的目標。此外，應用程式會將具有焦點的使用者介面元素反白。在 AIR for TV 應用程式中，管理焦點的方式不同於桌上型應用程式，原因如下：

- 桌上型應用程式通常使用 TAB 鍵將焦點切換至下一個使用者介面元素。使用 TAB 鍵並不適用於 AIR for TV 應用程式。遙控器裝置通常沒有 TAB 鍵。因此，並不適合像在桌上型電腦一樣使用 DisplayObject 的 tabEnabled 屬性來管理焦點。
- 桌上型應用程式通常需要使用者以滑鼠將焦點放在使用者介面元素上。

因此，請在應用程式中執行下列動作：

- 在舞台中新增事件偵聽程式來偵聽 Keyboard 事件，例如 KeyboardEvent.KEY_DOWN。
- 提供應用程式邏輯，來判斷要向使用者反白的使用者介面元素。請確定應用程式啟動後，會反白使用者介面元素。
- 根據應用程式邏輯，將舞台收到的 Keyboard 事件傳送給適當的使用者介面元素物件。

  您也可以使用 Stage.focus 或 Stage.assignFocus()，將焦點指定給使用者介面元素。然後，您可以在該 DisplayObject 中新增事件偵聽程式，以便接收鍵盤事件。

使用者介面設計

為了讓 AIR for TV 應用程式的使用者介面在電視上正常運作，請同時採納以下方面的建議：

- 應用程式的反應性
- 應用程式的易用性
- 使用者的個性與期望

反應性

使用下列秘訣，提高 AIR for TV 應用程式的反應性。

- 儘可能縮小應用程式的初始 SWF 檔。

  在初始 SWF 檔中，只載入啟動應用程式所需的資源。例如，只載入應用程式的啟動畫面影像。

  雖然這是對於桌上型 AIR 應用程式的建議，但對於 AIR for TV 裝置更為重要。其中一個原因是，AIR for TV 裝置並沒有相當於桌上型電腦的處理能力。此外，這些裝置是將應用程式儲存在 Flash 記憶體中，存取速度不如桌上型電腦的硬碟來得快。

- 使應用程式以每秒至少 20 格的影格速率執行。

  請將您的圖形設計成符合此目標。圖形作業的複雜度會影響每秒影格數。如需改善顯示效能的秘訣，請參閱最佳化 Adobe Flash Platform 的效能。

  備註：AIR for TV 裝置的圖形硬體通常以 60 Hz 或 120 Hz (每秒 60 或 120 次) 的頻率來更新螢幕。例如，為了在 60-Hz 或 120-Hz 的螢幕上顯示，硬體會以每秒 30 格或每秒 60 格的頻率掃描舞台是否有更新。不過，使用者是否體驗到這些較高的影格速率，需視應用程式圖形作業的複雜度而定。

- 在使用者輸入的 100 - 200 毫秒內更新螢幕。

  如果更新太久，使用者通常會變得不耐煩，而開始不斷按按鍵。

易用性

AIR for TV 應用程式的使用者是在「客廳」環境裡。他們會坐在房間裡電視機的另一頭，大約相距 10 英呎。房間有時很暗。他們通常使用遙控器裝置來輸入選擇。可能有多人同時使用應用程式，有時一起使用，有時輪流使用。

因此，若要設計方便在電視上使用的使用者介面，請注意下列幾點：

- 使用者介面元素要大。

  當設計文字、按鈕或其他任何使用者介面元素時，請考慮到使用者是坐在房間裡的另一頭。一切都要能夠在那樣的距離 (例如 10 英呎) 內輕鬆地觀看和閱讀。不要因為螢幕夠大就想要塞滿螢幕。

- 對比要夠，讓人能夠從房間裡的另一頭輕鬆地觀看和閱讀內容。
- 具有焦點的使用者介面元素要亮，才能明顯表示焦點在該元素。
- 必要時才使用動畫。例如，使用滑動效果來切換畫面可以有不錯的連貫效果。不過，如果動畫無助於使用者進行瀏覽，或不是應用程式固有的，反而會分散注意力。
- 務必在使用者介面上提供明顯的方式，讓使用者回到先前的畫面。

如需使用遙控器的詳細資訊，請參閱第 113 頁「遙控器輸入處理」。

使用者的個性與期望

AIR for TV 應用程式的使用者通常是要在輕鬆愉快的環境中，享受電視機娛樂。他們不見得具備電腦或技術方面的知識。

因此，請設計具有下列特色的 AIR for TV 應用程式：

- 不使用專業術語。
- 避免使用非回應不可的對話方塊。
- 不同於針對工作或技術環境，針對客廳環境要採用輕鬆而客氣的指示。
- 使用符合電視觀賞者預期的高播放品質圖形。
- 建立能輕鬆地與遙控器裝置一起使用的使用者介面。不要使用適用於桌上型或行動應用程式的使用者介面或設計元素。例如，桌上型和行動裝置上的使用者介面通常需要用滑鼠或手指點選及按一下按鈕。

字體及文字

您可以使用裝置字體或 AIR for TV 應用程式內嵌的字體。

裝置字體是安裝在裝置上的字體。所有 AIR for TV 裝置都有下列裝置字體：

字體名稱	說明
_sans	_sans 裝置字體是 sans-serif 字樣。_sans 裝置字體 (安裝在所有 AIR for TV 裝置上) 是 Myriad Pro。通常，在電視上 sans-serif 字樣看起來的效果優於 serif 字樣，這是檢視距離所造成的緣故。
_serif	_serif 裝置字體是 serif 字樣。所有 AIR for TV 裝置上都會安裝的 _serif 裝置字體是 Minion Pro。
_typewriter	_typewriter 裝置字體是等寬字體。所有 AIR for TV 裝置上都會安裝的 _typewriter 裝置字體是 Courier Std。

所有 AIR for TV 裝置也都有下列亞洲裝置字體：

字體名稱	語言	字樣類別	地區設定代碼
RyoGothicPlusN-Regular	日文	sans	ja
RyoTextPlusN-Regular	日文	serif	ja
AdobeGothicStd-Light	韓文	sans	ko
AdobeHeitiStd-Regular	簡體中文	sans	zh_CN
AdobeSongStd-Light	簡體中文	serif	zh_CN
AdobeMingStd-Light	繁體中文	serif	zh_TW 和 zh_HK

這些 AIR for TV 裝置字體具有下列特性：

- 來自 Adobe® Type Library
- 在電視上效果精緻
- 針對視訊字幕所設計
- 是外框字體，不是點陣圖字體

備註：裝置製造商通常會在裝置上安裝其他裝置字體。除了 AIR for TV 裝置字體，還會安裝這些由製造商提供的裝置字體。

Adobe 提供一個稱為 FontMaster Deluxe 的應用程式，可顯示裝置上的所有裝置字體。您可以從電視版 Flash Platform 取得此應用程式。

您也可以將字體內嵌於 AIR for TV 應用程式中。如需內嵌字體的詳細資訊，請參閱 「ActionScript 3.0 開發人員指南」的進階文字顯示。

關於使用 TLF 文字欄位，Adobe 有下列建議：

- 針對亞洲語言文字，請使用 TLF 文字欄位，以利用應用程式執行所在的地區設定。設定與 TLFTextField 物件相關之 TextLayoutFormat 物件的 locale 屬性。若要判斷目前的地區設定，請參閱 「ActionScript 3.0 開發人員指南」的選擇地區設定。
- 如果字體不是 AIR for TV 裝置字體，請在 TextLayoutFormat 物件的 fontFamily 屬性中指定字體名稱。如果裝置上有此字體，AIR for TV 就會使用此字體。如果裝置上沒有您要求的字體，AIR for TV 會根據 locale 設定，以適當的 AIR for TV 裝置字體替代。
- 在 fontFamily 屬性中指定 _sans、_serif 或 _typewriter，同時設定 locale 屬性，讓 AIR for TV 選擇正確的 AIR for TV 裝置字體。視地區而定，AIR for TV 會從本身的亞洲裝置字體組或非亞洲裝置字體組中選擇字體。這些設定提供簡單的方式，自動針對四大亞洲地區設定和英文使用正確字體。

備註：如果您針對亞洲語言文字使用傳統文字欄位，請指定 AIR for TV 裝置字體的字體名稱，以確保顯示正確。如果您知道目標裝置上還有安裝其他字體，也可以指定該字體。

關於應用程式效能，請注意下列幾點：

- 傳統文字欄位提供比 TLF 文字欄位更快的效能。
- 使用點陣圖字體的傳統文字欄位提供的效能最快。

  不同於外框字體提供每個字元的外框資料，點陣圖字體會提供每個字元的點陣圖。裝置字體和內嵌字體都可以是點陣圖字體。
- 如果您指定裝置字體，請確定目標裝置上已安裝該裝置字體。如果裝置上未安裝該裝置字體，AIR for TV 會尋找並使用裝置上安裝的其他字體。不過，此行為會降低應用程式的效能。

- 如同於所有顯示物件，如果 TextField 物件大多不變，請將物件的 cacheAsBitmap 屬性設為 true。此設定可改善轉場 (如淡入、滑動及 Alpha 混合) 的效能。使用 cacheAsBitmapMatrix 進行縮放和轉譯。如需詳細資訊，請參閱第 108 頁 「圖形硬體加速」。

檔案系統安全性

AIR for TV 應用程式是 AIR 應用程式，因此可以存取裝置的檔案系統。不過，在「客廳」裝置中，應用程式不能存取裝置的系統檔案或其他應用程式的檔案，這一點非常重要。電視和相關裝置的使用者不能容忍任何裝置失靈情形 — 畢竟他們是在看電視。

因此，AIR for TV 應用程式只提供裝置檔案系統的有限檢視。使用 ActionScript 3.0 時，應用程式只能存取特定目錄 (和其子目錄)。此外，您在 ActionScript 中使用的目錄名稱並不是裝置上的實際目錄名稱。這樣額外一層的保護，可避免 AIR for TV 應用程式惡意或意外存取不屬於它們的本機檔案。

如需詳細資訊，請參閱 AIR for TV 應用程式的目錄檢視。

AIR 應用程式執行程序

AIR for TV 應用程式會在 AIR 應用程式執行程序內執行，如 AIR 應用程式執行程序中所描述。

對於 AIR for TV 應用程式而言，唯一的差別在於它們對檔案系統的存取有限，如第 117 頁 「檔案系統安全性」中所描述。

應用程式效期

不同於桌上型電腦環境，使用者無法關閉 AIR for TV 應用程式的執行視窗。因此，請提供使用者介面機制來結束應用程式。

通常，裝置可讓使用者透過遙控器的結束按鍵，無條件結束應用程式。不過，AIR for TV 不會將事件 flash.events.Event.EXITING 傳送至應用程式。因此，請經常儲存應用程式狀態，讓應用程式在下一次啟動時可以恢復到合理狀態。

HTTP Cookie

AIR for TV 支援 HTTP 永續性 Cookie 與工作階段 Cookie。AIR for TV 會將每一個 AIR 應用程式的 Cookie 儲存於特定應用程式目錄：

```
/app-storage/<app id>/Local Store
```

Cookie 檔案的名稱是 cookies。

備註：其他裝置上的 AIR，例如桌上型裝置，不會分開儲存每個應用程式的 Cookie。特定應用程式 Cookie 儲存區會支援 AIR for TV 的應用程式和系統安全性模型。

以下列方式使用 ActionScript 屬性 URLRequest.manageCookies：

- 將 manageCookies 設為 true。此值為預設值。這表示 AIR for TV 會自動將 Cookie 增加至 HTTP 要求，並記住 HTTP 回應中的 Cookie。

  備註：即使 manageCookies 為 true，應用程式也可以使用 URLRequest.requestHeaders，手動將 Cookie 新增至 HTTP 要求。如果此 Cookie 的名稱與 AIR for TV 管理的 Cookie 相同，則要求會包含兩個同名的 Cookie。這兩個 Cookie 的值可能不同。

- 將 manageCookies 設為 false。此值表示應用程式負責在 HTTP 要求中傳送 Cookie，並記住 HTTP 回應中的 Cookie。

如需詳細資訊，請參閱 URLRequest。

開發 AIR for TV 應用程式的工作流程

您可以利用下列 Adobe Flash Platform 開發工具來開發 AIR for TV 應用程式：

- Adobe Flash Professional

  Adobe Flash Professional CS5.5 是第一個支援 AIR for TV 的應用程式，其支援 AIR 2.5 for TV。

- Adobe Flash® Builder®

  Flash Builder 4.5 支援 AIR 2.5 for TV。

- AIR SDK

  自 AIR 2.5 起，您可以使用 AIR SDK 隨附的命令列工具來開發應用程式。若要下載 AIR SDK，請參閱 http://www.adobe.com/tw/products/air/sdk/。

使用 Flash Professional

使用 Flash Professional 開發、測試及發佈 AIR for TV 應用程式的方式，類似於使用 AIR 桌上型應用程式的工具。

不過，在撰寫 ActionScript 3.0 程式碼時，請僅使用 tv 和 extendedTV AIR 描述檔支援的類別和方法。如需詳細資料，請參閱第 211 頁 「裝置描述檔」。

專案設定

請執行下列動作，設定 AIR for TV 應用程式的專案：

- 在「發佈設定」對話方塊的「Flash」索引標籤中，將「播放程式」值至少設為 AIR 2.5。
- 在「Adobe AIR 設定」(應用程式和安裝程式設定) 對話方塊的 「一般」索引標籤中，將描述檔設為 TV 或延伸 TV。

除錯

您可以在 Flash Professional 內使用 AIR Debug Launcher 來執行應用程式。請執行下列動作：

- 若要以除錯模式執行應用程式，請選取：

  「除錯 > 影片除錯 > 在 AIR Debug Launcher (桌面) 中」

  完成此選擇後，後續再執行除錯時，您可以選取：

  「除錯 > 影片除錯 > 除錯」

- 若要執行應用程式而不使用除錯模式功能，請選取：

  「控制 > 測試影片 > 在 AIR Debug Laucher (桌面) 中」

  完成此選擇後，後續再執行時，您可以選取 「控制 > 測試影片 > 測試」。

因為您將 AIR 描述檔設為 TV 或延伸 TV，AIR Debug Launcher 會提供名為 「遙控器按鈕」的選單。您可以使用此選單來模擬按下遙控器裝置上的按鍵。

如需詳細資訊，請參閱第 125 頁 「使用 Flash Professional 遠端除錯」。

使用原生擴充功能

如果應用程式使用原生擴充功能，請依照第 130 頁 「使用原生擴充功能的工作清單」中的指示執行。

不過，當應用程式使用原生擴充功能時：

- 您無法使用 Flash Professional 來發佈應用程式。若要發佈應用程式，請使用 ADT。請參閱第 123 頁 「使用 ADT 封裝」。

- 您無法執行應用程式，或使用 Flash Professional 來進行應用程式除錯。若要在開發電腦上進行應用程式除錯，請使用 ADL。請參閱第 124 頁 「使用 ADL 進行裝置模擬」。

使用 Flash Builder

自 Flash Builder 4.5 起，Flash Builder 支援 AIR for TV 的開發。使用 Flash Builder 開發、測試及發佈 AIR for TV 應用程式的方式，類似於使用 AIR 桌上型應用程式的工具。

設定應用程式

請確定應用程式：

- 在 MXML 檔中使用 Application 元素作為容器類別 (如果您使用 MXML 檔)：

```
<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>.
```

  重要事項：AIR for TV 應用程式不支援 WindowedApplication 元素。

  備註：您完全不需要使用 MXML 檔。您可以改為建立 ActionScript 3.0 專案。

- 僅使用 tv 和 extendedTV AIR 描述檔支援的 ActionScript 3.0 類別和方法。如需詳細資料，請參閱第 211 頁 「裝置描述檔」。

另外在應用程式 XML 檔中，請確定：

- application 元素的 xmlns 屬性設為 AIR 2.5：

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

- supportedProfiles 元素包含 tv 或 extendedTV：

```
<supportedProfiles>tv</supportedProfiles>
```

應用程式除錯

您可以在 Flash Builder 內使用 AIR Debug Launcher 來執行應用程式。請執行下列動作：

1. 選取 「執行 > 除錯組態」。
2. 確定 「描述檔」欄位設為 「桌上型」。
3. 選取 「執行 > 除錯」以用除錯模式執行應用程式，或選取 「執行 > 執行」以執行應用程式但不使用除錯模式功能。

因為您將 supportedProfiles 元素設為 TV 或延伸 TV，AIR Debug Launcher 會提供名為 「遙控器按鈕」的選單。您可以使用此選單來模擬按下遙控器裝置上的按鍵。

如需詳細資訊，請參閱第 126 頁 「使用 Flash Builder 遠端除錯」。

使用原生擴充功能

如果應用程式使用原生擴充功能，請依照第 130 頁 「使用原生擴充功能的工作清單」中的指示執行。

不過，當應用程式使用原生擴充功能時：

- 您無法使用 Flash Builder 來發佈應用程式。若要發佈應用程式，請使用 ADT。請參閱第 123 頁 「使用 ADT 封裝」。
- 您無法執行應用程式，或使用 Flash Builder 來進行應用程式除錯。若要在開發電腦上進行應用程式除錯，請使用 ADL。請參閱第 124 頁 「使用 ADL 進行裝置模擬」。

AIR for TV 應用程式描述器屬性

如同其他 AIR 應用程式，設定應用程式描述器檔案中的基本應用程式屬性。電視描述檔應用程式會忽略某些桌上型特定的屬性，像是視窗大小與透明度。extendedTV 描述檔中的應用程式目標裝置可以使用原生擴充功能。這些應用程式會識別 extensions 元素中使用的原生擴充功能。

一般設定

對於所有電視描述檔應用程式，有數個重要的應用程式描述器設定。

必要的 AIR 執行階段版本

使用應用程式描述器檔案的命名空間，指定應用程式所需的 AIR 執行階段版本。

application 元素中指派的命名空間，大致上決定了應用程式可使用的功能。例如，假設應用程式使用 AIR 2.5 命名空間，但使用者已安裝某個較新的版本。在此情況下，即使在較新的 AIR 版本中有不同的行為，應用程式看到的仍是 AIR 2.5 行為。只有當您變更命名空間並發佈更新時，您的應用程式才能存取新的行為和功能。安全性修正為此規則的重要例外。

使用 application 根元素的 xmlns 屬性，指定命名空間：

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

AIR 2.5 是第一個支援電視應用程式的 AIR 版本。

應用程式身分識別

針對發佈的每個應用程式，有數種設定必須是唯一的。這些設定包括 id、name 和 filename 元素。

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

應用程式版本

在 versionNumber 元素中指定應用程式版本。指定 versionNumber 的值時，可以使用最多三個號碼的序列，號碼之間以點分隔，例如：“0.1.2”。版本號碼的每個區段最多可以有三位數字。(亦即，“999.999.999” 為允許的最大版本號碼)。號碼中毋須包括所有三個區段；“1” 與 “1.0” 都是合法的版本號碼。

您也可以使用 versionLabel 元素，指定版本的標籤。新增版本標籤時，會顯示標籤，而不是版本號碼。

```
<versionNumber>1.23.7<versionNumber>
<versionLabel>1.23 Beta 7</versionLabel>
```

主要應用程式 SWF

在 initialWindow 元素的 versionLabel 子元素中，指定主要應用程式 SWF 檔案。以電視描述檔中的裝置為目標時，必須使用 SWF 檔案 (不支援 HTML 類型應用程式)。

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

您必須將檔案包括在 AIR 套件中 (使用 ADT 或您的 IDE)。只是參照應用程式描述器中的名稱，不會將檔案自動包括在套件中。

主要螢幕屬性

initialWindow 元素中有數個子元素控制著應用程式主畫面的初始外觀與行為。雖然電視描述檔會在裝置上忽略這其中大部分屬性，但您可以使用 fullScreen 元素：

- fullScreen — 指定應用程式是應佔滿整個裝置顯示器，還是與一般作業系統顏色共用顯示器。

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

visible 元素

visible 元素是 initialWindow 元素的子元素。AIR for TV 會忽略 visible 元素，因為在 AIR for TV 裝置上您的應用程式內容永遠為可見的。

不過，如果您的應用程式也以桌上型裝置為目標，請將 visible 元素設定為 true。

在桌上型裝置上，這個元素的值預設為 false。因此，如果未包含 visible 元素，在桌上型裝置上就看不到應用程式的內容。雖然您可以使用 ActionScript 類別 NativeWindow 讓桌上型裝置顯示內容，但是電視裝置描述檔不支援 NativeWindow 類別。如果您嘗試在執行 AIR for TV 裝置的應用程式上使用 NativeWindow 類別，應用程式將無法載入。無論您是否呼叫 NativeWindow 類別的方法，使用該類別的應用程式都無法在 AIR for TV 裝置上載入。

支援的描述檔

如果應用程式只適合在電視裝置上執行，則您可以防止應用程式安裝到其他類型的電腦裝置上。請在 supportedProfiles 元素排除支援清單中的其他描述檔：

```
<supportedProfiles>tv extendedTV</supportedProfiles>
```

如果應用程式會使用原生擴充功能，請只在支援描述檔清單中包含 extendedTV 描述檔：

```
<supportedProfiles>extendedTV</supportedProfiles>
```

如果省略 supportedProfiles 元素，則應用程式會假設支援所有描述檔。

不要「只」在 supportedProfiles 清單中包含 tv 描述檔。有些電視裝置會永遠以對應於 extendedTV 描述檔的模式執行 AIR for TV。這個行為使得 AIR for TV 可以執行使用原生擴充功能的應用程式。如果您的 supportedProfiles 元素只指定 tv，即表示您的內容與 extendedTV 的 AIR for TV 模式不相容。因此，有些電視裝置不會載入只指定 tv 描述檔的應用程式。

如需 tv 和 extendedTV 設定檔中支援之 ActionScript 類別的清單，請參閱第 212 頁 「不同描述檔的功能」。

必須要有原生擴充功能

支援 extendedTV 描述檔的應用程式可以使用原生擴充功能。

請使用 extensions 和 extensionID 元素，在應用程式描述器中宣告 AIR 應用程式會使用的所有原生擴充功能。下列範例說明指定兩個必要原生擴充功能的語法：

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

如果擴充功能未列出，應用程式便無法加以使用。

extensionID 元素的值與擴充功能描述器檔案中的 id 元素相同。擴充功能描述器檔案是一個名為 extension.xml 的 XML 檔。它封裝在您自裝置製造商得到的 ANE 檔中。

如果您列出 extensions 元素中的擴充功能，但 AIR for TV 裝置未安裝該擴充功能，應用程式將無法執行。這個規則的例外是，在使用 AIR for TV 應用程式封裝的 ANE 檔案具有虛設常式版擴充功能的情況下。在此情況下，應用程式可以執行，但使用的是虛設常式版擴充功能。虛設常式版本具有 ActionScript 程式碼，但沒有原生程式碼。

應用程式圖示

在電視裝置中使用應用程式圖示的需求依裝置而異。例如，裝置製造商會指定：

- 必要的圖示和圖示大小。
- 必要的檔案類型和命名慣例。
- 提供應用程式圖示的方式，例如是否將圖示封裝在應用程式中。
- 是否在應用程式描述器檔案的 icon 元素中指定圖示。
- 應用程式未提供圖示時的行為。

如需詳細資訊，請洽詢裝置製造商。

忽略的設定

電視裝置上的應用程式將忽略適用於行動、原生視窗或桌上型作業系統功能的應用程式設定。忽略的設定如下：

- allowBrowserInvocation
- aspectRatio
- autoOrients
- customUpdateUI
- fileTypes
- height
- installFolder
- maximizable
- maxSize
- minimizable
- minSize
- programMenuFolder
- renderMode
- resizable
- systemChrome
- title
- transparent
- visible
- width
- x
- y

封裝 AIR for TV 應用程式

使用 ADT 封裝

您可以使用 AIR ADT 命令列工具來封裝 AIR for TV 應用程式。自 AIR SDK 版本 2.5 起，ADT 支援為電視裝置封裝。在封裝之前，請編譯所有的 ActionScript 和 MXML 程式碼。您也必須具有程式碼簽署憑證。您可以使用 ADT -certificate 命令建立憑證。

如需 ADT 命令與選項的詳細參考資訊，請參閱 第 142 頁 「AIR Developer Tool (ADT)」。

建立 AIR 套件

若要建立 AIR 套件，請使用 ADT package 命令：

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

此範例假設：

- ADT 工具的路徑位於命令列殼層的路徑定義中 (請參閱第 263 頁 「Path 環境變數」)。
- codesign.p12 憑證是您執行 ADT 命令時所在位置的父目錄。

請從包含應用程式檔案的目錄中執行命令。範例中的應用程式檔案為 myApp-app.xml (應用程式描述器檔案)、myApp.swf 與圖示目錄。

當您執行範例所示的命令時，ADT 會提示您輸入金鑰儲存密碼。不是所有的殼層程式都會顯示您輸入的密碼字元；輸入完成時按下 Enter 即可。或者，您也可以在 ADT 命令中使用 storepass 參數來包含密碼。

建立 AIRN 套件

如果 AIR for TV 應用程式會使用原生擴充功能，請建立 AIRN 套件，而非 AIR 套件。若要建立 AIRN 套件，請使用 ADT package 命令，並將目標類型設為 airn。

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp-app.xml myApp.swf
icons -extdir C:\extensions
```

此範例假設：

- ADT 工具的路徑位於命令列殼層的路徑定義中 (請參閱第 263 頁 「Path 環境變數」)。
- codesign.p12 憑證是您執行 ADT 命令時所在位置的父目錄。
- -extdir 參數指出應用程式使用之 ANE 檔案所在的目錄。

  這些 ANE 檔案包含僅有 ActionScript 虛設常式，或擴充功能的模擬版。AIR for TV 裝置上安裝的是含原生程式碼的擴充功能版本。

請從包含應用程式檔案的目錄中執行命令。範例中的應用程式檔案為 myApp-app.xml (應用程式描述器檔案)、myApp.swf 與圖示目錄。

當您執行範例所示的命令時，ADT 會提示您輸入金鑰儲存密碼。不是所有的殼層程式都會顯示您輸入的密碼字元；輸入完成時按下 Enter 即可。或者，您也可以在命令中使用 storepass 參數來包含密碼。

您也可以為使用原生擴充功能的 AIR for TV 應用程式建立 AIRI 檔。AIRI 檔就像 AIRN 檔一樣，差別在於未經簽署。例如：

```
adt -prepare myApp.airi myApp.xml myApp.swf icons -extdir C:\extensions
```

當您準備好要簽署應用程式時，便可以從 AIRI 檔建立 AIRN 檔：

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp.airi
```

如需詳細資訊，請參閱開發 Adobe AIR 的原生擴充功能。

使用 Flash Builder 或 Flash Professional 封裝

Flash Professional 與 Flash Builder 讓您毋須自行執行 ADT，便能發佈或匯出 AIR 套件。在這些程式的文件中，涵蓋為 AIR 應用程式建立 AIR 套件的程序。

但是，目前僅有 ADT 可建立 AIRN 套件，這是使用原生擴充功能之 AIR for TV 應用程式的應用程式套件。

如需詳細資訊，請參閱下列章節：

- 使用 Flash Builder 封裝 AIR 應用程式
- 使用 Flash Professional 發佈 Adobe AIR

AIR for TV 應用程式的除錯

使用 ADL 進行裝置模擬

測試與除錯多數應用程式功能的最快、最簡單的方法，便是使用 Adobe Debug Launcher (ADL) 公用程式，在您的開發電腦上執行應用程式。

ADL 使用應用程式描述器中的 supportedProfiles 元素，選擇要使用的描述檔。明確說明如下：

- 若列出超過一個以上的描述檔，ADL 會使用清單中的第一個。
- 您可以使用 ADL 的 -profile 參數，再選取 supportedProfiles 清單內的另一個描述檔。
- 如果應用程式描述器中未包含 supportedProfiles 元素，則您可以為 -profile 引數指定任何描述檔。

例如，使用下列命令可啟動應用程式，以模擬 tv 描述檔：

```
adl -profile tv myApp-app.xml
```

以 ADL 在桌上型電腦上模擬 tv 或 extendedTV 描述檔時，應用程式執行的環境更符合目標裝置。例如：

- 不在 -profile 引數描述檔中的 ActionScript API 將無法使用。
- ADL 允許透過選單命令接收裝置輸入控制器 (如遙控器) 的輸入。
- 在 -profile 引數中指定 tv 或 extendedTV，可讓 ADL 在桌上型電腦上模擬 StageVideo 類別。
- 在 -profile 引數中指定 extendedTV，可讓應用程式使用應用程式 AIRN 檔內封裝的原生擴充功能虛設常式或模擬程式。

不過，由於 ADL 在桌上型電腦上執行應用程式，使用 ADL 測試 AIR for TV 應用程式有其限制：

- 它不會反映裝置上的應用程式效能。在目標裝置上執行效能測試。
- 它不會模擬 StageVideo 物件的限制。一般而言，以 AIR for TV 裝置為目標時，會使用 StageVideo 類別 (而非 Video 類別) 來播放視訊。StageVideo 類別會利用裝置硬體的效能優勢，但是有顯示限制。ADL 在桌上型電腦上播放視訊時，沒有這些限制。因此，請在目標裝置上測試播放視訊。
- 它無法模擬原生擴充的原生程式碼。不過，您可以在 ADL -profile 引數中，指定支援原生擴充功能的 extendedTV 描述檔。ADL 可讓您使用僅有 ActionScript 的虛設常式，或 ANE 套件 中擴充功能的模擬版來測試。不過，裝置上安裝的對應擴充功能通常還包含原生程式碼。若要使用含原生程式碼的擴充功能進行測試，請在目標裝置上執行應用程式。

如需詳細資訊，請參閱 第 137 頁 「AIR Debug Launcher (ADL)」。

使用原生擴充功能

如果應用程式會使用原生擴充功能，則 ADL 命令看起來會如同下列範例：

```
adl -profile extendedTV -extdir C:\extensionDirs myApp-app.xml
```

此範例假設：

- ADL 工具的路徑位於命令列殼層的路徑定義中 (請參閱第 263 頁 「Path 環境變數」)。
- 目前的目錄包含應用程式檔案。這些檔案包括 SWF 檔和應用程式描述器檔案 (即此範例中的 myApp-app.xml)。
- 參數 -extdir 會命名目錄，此目錄包含應用程式使用之每一個原生擴充功能的目錄。每個目錄都包含原生擴充功能的 「未封裝」 ANE 檔案。例如：

```
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
```

這些未封裝的 ANE 檔案包含僅有 ActionScript 虛設常式，或擴充功能的模擬版。AIR for TV 裝置上安裝的是含原生程式碼的擴充功能版本。

如需詳細資訊，請參閱開發 Adobe AIR 的原生擴充功能。

遙控器輸入

ADL 會模擬電視裝置上的遙控器按鈕。當使用電視描述檔啟動 ADL 時，您可以使用顯示的選單，傳送這些按鈕輸入至模擬的裝置。

螢幕大小

您可以設定 ADL -screensize 參數，以便在不同大小的螢幕上測試您的應用程式。您可以指定包含四個值的字串，這四個值分別代表一般與最大螢幕的寬度與高度。

請一律指定縱向版面的像素尺寸，也就是將寬度的值指定為小於高度的值。例如：

```
adl -screensize 728x1024:768x1024 myApp-app.xml
```

追蹤陳述式

在桌上型電腦上執行您的電視應用程式時，追蹤輸出會列印至除錯程式或用來啟動 ADL 的終端機視窗。

使用 Flash Professional 遠端除錯

您可以使用 Flash Professional，對目標裝置上執行的 AIR for TV 應用程式進行遠端除錯。不過，設定遠端除錯的步驟需視裝置而定。例如，Adobe® AIR® for TV MAX 2010 硬體開發套件即包含文件來說明該裝置適用的詳細步驟。

不過，無論目標裝置為何，請執行下列步驟來準備進行遠端除錯：

1 在「發佈設定」對話方塊的「Flash」索引標籤中，選取「允許除錯」。

此選項會讓 Flash Professional 在從 FLA 檔建立的所有 SWF 檔中包含除錯資訊。

2 在「Adobe AIR 設定」對話方塊 (應用程式和安裝程式設定) 的「簽名」索引標籤中，選取準備 AIR 中間 (AIRI) 檔的選項。

當您還在開發應用程式時，使用不需要數位簽名的 AIRI 檔案便已足夠。

3 發佈應用程式並建立 AIRI 檔。

最後的步驟是在目標裝置上安裝並執行應用程式。不過，這些步驟視裝置而定。

使用 Flash Builder 遠端除錯

您也可以使用 Flash Builder，對目標裝置上執行的 AIR for TV 應用程式進行遠端除錯。不過，執行遠端除錯的步驟視裝置而定。

不過，無論目標裝置為何，請執行下列步驟來準備進行遠端除錯：

1 選取「專案 > 匯出發行組建」。選取準備 AIR 中間 (AIRI) 檔的選項。

當您還在開發應用程式時，使用不需要數位簽名的 AIRI 檔案便已足夠。

2 發佈應用程式並建立 AIRI 檔。

3 變更應用程式的 AIRI 套件以納入含有除錯資訊的 SWF 檔。

含有除錯資訊的 SWF 檔是位於應用程式的 Flash Builder 專案目錄中一個名為 bin-debug 的目錄底下。請將 AIRI 套件中的 SWF 檔取代為 bin-debug 目錄中的 SWF 檔。

在 Windows 開發電腦上，您可以執行下列動作來完成此取代工作：

1 將 AIRI 套件檔案的副檔名部分重新命名為 .zip，而非 .airi。

2 解壓縮 ZIP 檔案內容。

3 以 bin-debug 的 SWF 檔案，取代解壓縮的目錄結構中的 SWF 檔案。

4 將解壓縮的目錄中的檔案重新壓縮。

5 再次將壓縮檔案的副檔名變回 .airi。

如果您使用 Mac 開發電腦，則此取代工作的步驟視裝置而定。不過，這些步驟通常包括：

1 在目標裝置上安裝 AIRI 套件。

2 將目標裝置上應用程式安裝目錄中的 SWF 檔，取代為 bin-debug 目錄中的 SWF 檔。

例如，以 Adobe AIR for TV MAX 2010 硬體開發套件隨附的裝置為例。依套件文件所述，安裝 AIRI 套件。然後在 Mac 開發電腦的命令列中，使用 telnet 存取目標裝置。以 bin-debug 目錄中的 SWF，取代應用程式安裝目錄中位於 /opt/adobe/stagecraft/apps/< 應用程式名稱 >/ 中的 SWF 檔。

下列步驟適用於使用 Flash Builder 以及 Adobe AIR for TV MAX 2010 硬體開發套件隨附的裝置進行遠端除錯。

1 在執行 Flash Builder 的電腦 (開發電腦) 上，執行 MAX 2010 硬體開發套件隨附的 AIR for TV Device Connector。它會顯示開發電腦的 IP 位址。

2 在硬體套件裝置上，啟動開發套件同時還隨附的 DevMaster 應用程式。

3 在 DevMaster 應用程式中，輸入 AIR for TV Device Connector 所顯示您開發電腦的 IP 位址。

4 在 DevMaster 應用程式中，確定已選取「啟用遠端除錯」。

5 結束 DevMaster 應用程式。

6 在開發電腦上，選取 AIR for TV Connector 中的「啟動」。

7 在硬體套件裝置上，啟動另一個應用程式。確認 AIR for TV Device Connector 中顯示了追蹤資訊。

如果未顯示追蹤資訊，表示開發電腦和硬體套件裝置未連線。確定開發電腦上用於追蹤資訊的連接埠可以使用。您可以在 AIR for TV Device Connector 中選擇不同的連接埠。另外，請確定防火牆允許他人存取所選連接埠。

接下來，在 Flash Builder 中啟動除錯程式。請執行下列動作：

1 在 Flash Builder 中，選取「Run > Debug Configurations」。

2 從用於進行本機除錯的現有除錯組態中，複製專案的名稱。

3 在「Debug Configurations」對話方塊中，選取「Web Application」。然後選取「New Launch Configuration」圖示。

4 將專案名稱貼到「Project」欄位中。

5 在「URL Or Path To Launch」區段中，取消勾選「Use Default」。另外，在文字欄位中輸入 **about:blank**。

6 選取「Apply」以儲存變更。

7 選取「Debug」以啟動 Flash Builder 除錯程式。

8 在硬體套件裝置上啟動應用程式。

現在，您可以使用 Flash Builder 除錯程式來設定中斷點和檢查變數 (舉例來說)。

第 9 章 使用 Adobe AIR 的原生擴充功能

Adobe AIR 原生擴充功能的 ActionScript API，可讓您存取以原生程式碼進行程式設計的特定裝置功能。原生擴充功能開發人員有時候會與裝置製造商合作，有時候則成為協力廠商開發人員。

如果您正在開發原生擴充功能，請參閱 Developing Native Extensions for Adobe AIR。

原生擴充功能是以下的組合：

- ActionScript 類別。
- 原生程式碼。

但是，您身為使用原生擴充功能的 AIR 應用程式開發人員，僅能使用 ActionScript 類別。

下列情況中，原生擴充功能有用：

- 原生程式碼實作可讓您存取平台特定的功能。內建 ActionScript 類別中沒有這些平台特定的功能，而且這些功能也無法在應用程式特定的 ActionScript 類別中實作。原生程式碼實作可提供這樣的功能，因為它可讓您存取特定裝置硬體和軟體。
- 原生程式碼實作有時候可能會比只使用 ActionScript 的實作更快。
- 原生程式碼實作可讓您存取舊版原生程式碼的 ActionScript。

Adobe 開發人員中心提供一些原生擴充功能的範例。例如，有一種原生擴充功能可讓 AIR 應用程式存取 Android 震動功能。請參閱 Native extensions for Adobe AIR。

AIR 原生擴充功能 (ANE) 檔案

原生擴充功能開發人員會將原生擴充功能封裝成 ANE 檔案。ANE 檔案是包含原生擴充功能必要元件庫與資源的封存檔案。

請注意，在某些裝置中，ANE 檔案包含原生擴充功能使用的原生程式碼元件庫。但在其他裝置上，原生程式碼元件庫是安裝於裝置。在某些案例中，原生擴充功能完全沒有特定裝置的原生程式碼；僅以 ActionScript 來實作。

您身為 AIR 應用程式開發人員，應依照以下方式使用 ANE 檔案：

- 使用相同於在元件庫路徑中包含 SWC 檔案的方式，在應用程式元件庫路徑中包含 ANE 檔案。此動作可讓應用程式參考擴充功能的 ActionScript 類別。

  備註：編譯您的應用程式時，請務必使用 ANE 的動態連結。如果使用 Flash Builder，請在 ActionScript Builder 路徑屬性面板上指定「外部」；如果使用命令列，則指定 -external-library-path。
- 以 AIR 應用程式封裝 ANE 檔案。

原生擴充功能與 NativeProcess ActionScript 類別的比較

ActionScript 3.0 提供 NativeProcess 類別。此類別可以讓 AIR 應用程式在主機作業系統上執行原生處理程序。此功能與原生擴充功能相似，可讓您存取特定平台功能與元件庫。比較並決定使用 NativeProcess 類別或原生擴充功能時，請考量以下幾點：

- 僅有 extendedDesktop AIR 描述檔支援 NativeProcess 類別。因此，在 AIR 描述檔為 mobileDevice 和 extendedMobileDevice 的應用程式中，原生功能擴充是唯一的選擇。
- 原生擴充功能開發人員經常提供不同平台的原生實作，但是他們提供的 ActionScript API，通常是跨平台皆相同。使用 NativeProcess 類別時，視不同的平台而定，要啟動原生處理程序的 ActionScript 程式碼也可能有所差異。

- NativeProcess 類別會啟動個別處理程序，而原生擴充功能會在與 AIR 應用程式相同的處理程序中執行。因此，如果您擔心程式碼當掉，使用 NativeProcess 類別比較保險。但是，將程序分開，表示您可能要實作內部程序通訊處理。

原生擴充功能與 ActionScript 類別元件庫 (SWC 檔案) 的比較

SWC 檔案是封存格式的 ActionScript 類別元件庫。SWC 檔案包含 SWF 檔案和其他資源檔案。SWC 檔案是共用 ActionScript 類別的便利方式，而不需要共用個別的 ActionScript 程式碼與資源檔案。

原生擴充功能套件是 ANE 檔案。正如 SWC 檔案，ANE 檔案也是 ActionScript 類別元件庫，包含 SWF 檔案和其他封存格式的資源檔案。但是，AME 檔案與 SWC 檔案間最重要的差異是，僅有 ANE 檔案可包含原生程式碼元件庫。

備註：編譯您的應用程式時，請務必使用 ANE 檔案的動態連結。如果使用 Flash Builder，請在 ActionScript Builder 路徑屬性面板上指定 「外部」；如果使用命令列，則指定 -external-library-path。

更多說明主題

關於 SWC 檔案

支援的裝置

自 AIR 3 起，您可在下列裝置中使用應用程式的原生擴充功能：

- Android 裝置 (Android 2.2 以上)
- iOS 裝置 (iOS 4.0 以上)
- iOS 模擬器 (AIR 3.3 以上)
- Blackberry PlayBook
- 支援 AIR 3.0 的 Windows 桌上型裝置
- 支援 AIR 3.0 的 Mac OS X 桌上型裝置

通常，相同的原生擴充功能目標是多個平台。擴充功能的 ANE 檔案包含每個支援平台的 ActionScript 與原生元件庫。通常，在所有平台上，ActionScript 元件庫都有相同的公開介面。原生元件庫必須不同。

有時候原生擴充功能會支援預設平台。預設平台實作僅有 ActionScript 程式碼，但是沒有原生程式碼。如果您在擴充功能未特別支援的平台上封裝應用程式，則應用程式執行時會使用預設實作。例如，考量提供僅套用至行動裝置之功能的擴充功能。擴充功能也可提供桌上型應用程式可用於模擬功能的預設實作。

支援的裝置描述檔

下列 AIR 描述檔支援原生擴充功能：

- extendedDesktop，自 AIR 3.0 起
- mobileDevice，自 AIR 3.0 起
- extendedMobileDevice，自 AIR 3.0 起

更多說明主題

AIR 描述檔支援

使用原生擴充功能的工作清單

若要在您的應用程式中使用原生擴充功能，請執行下列工作：

1 在您的應用程式描述器檔案中宣告擴充功能。

2 在應用程式元件庫路徑中包含 ANE 檔案。

3 封裝應用程式。

在您的應用程式描述器檔案中宣告擴充功能

所有 AIR 應用程式都有應用程式描述器檔案。當應用程式使用原生擴充功能時，應用程式描述器檔案會包含 <extensions> 元素。例如：

```
<extensions>
    <extensionID>com.example.Extension1</extensionID>
    <extensionID>com.example.Extension2</extensionID>
</extensions>
```

extensionID 元素的值與擴充功能描述器檔案中的 id 元素相同。擴充功能描述器檔案是一個名為 extension.xml 的 XML 檔。它是封裝於 ANE 檔案。您可使用封存解壓縮工具來查看 extension.xml 檔案。

在應用程式元件庫路徑中包含 ANE 檔案

若要編譯使用原生擴充功能的應用程式，請在元件庫路徑中包含 ANE 檔案。

與 Flash Builder 搭配使用的 ANE 檔案

如果應用程式使用原生擴充功能，請在元件庫路徑中包含原生擴充功能的 ANE 檔案。然後，您可以使用 Flash Builder 來編譯 ActionScript 程式碼。

請使用 Flash Builder 4.5.1 執行下列步驟：

1 將 ANE 檔案的副檔名 .ane 變更為 .swc。這是讓 Flash Builder 找到檔案的必要步驟。

2 在 Flash Builder 專案上選取 「專案 > 屬性」。

3 選取 「屬性」對話方塊中的 「Flex 建置路徑」。

4 在 「元件庫路徑」索引標籤中，選取 「新增 SWC...」。

5 瀏覽至 SWC 檔並選取 「開啟」。

6 選取 「新增 SWC...」對話方塊中的 「確定」。

現在，ANE 檔案會出現在 「屬性」對話方塊的 「元件庫路徑」索引標籤中。

7 展開 SWC 檔項目。按兩下 「連結類型」，以開啟 「元件庫路徑項目選項」對話方塊。

8 在 「元件庫路徑項目選項」對話方塊中，將 「連結類型」變更為 「外部」。

現在，您可以使用 「專案 > 建置專案」來編譯應用程式 (舉例來說)。

與 Flash Professional 搭配使用 ANE 檔案

如果應用程式使用原生擴充功能，請在元件庫路徑中包含原生擴充功能的 ANE 檔案。然後，您可以使用 Flash Professional CS5.5 來編譯 ActionScript 程式碼。請執行下列動作：

1. 將 ANE 檔案的副檔名 .ane 變更為 .swc。這是讓 Flash Professional 找到檔案的必要步驟。
2. 在 FLA 檔上選取「檔案 > ActionScript 設定」。
3. 在「進階 ActionScript 3.0 設定」對話方塊中選取「元件庫路徑」索引標籤。
4. 選取「瀏覽至 SWC 檔案」按鈕。
5. 瀏覽至 SWC 檔並選取「開啟」。

   現在，SWC 檔出現在「進階 ActionScript 3.0 設定」對話方塊的「元件庫路徑」索引標籤中。
6. 選取 SWC 檔後，選取「選取元件庫的連結選項」按鈕。
7. 在「元件庫路徑項目選項」對話方塊中，將「連結類型」變更為「外部」。

封裝使用原生擴充功能的應用程式

使用 ADT 來封裝使用原生擴充功能的應用程式。您無法使用 Flash Professional CS5.5 或 Flash Builder 4.5.1 封裝應用程式。

使用 ADT 的詳細資訊在 AIR Developer Tool (ADT)。

例如，下列 ADT 命令會建立使用原生擴充功能之應用程式適用的 DMG 檔案 (Mac OS X 的原生安裝程式檔案)：

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

下列命令會建立 Android 裝置的 APK 套件。

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

下列命令會建立 iPhone 應用程式的 iOS 套件。

```
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
```

請注意以下各點：

- 使用原生安裝程式套件類型。

- 指定擴充功能目錄。
- 請確定 ANE 檔案支援應用程式的目標裝置。

使用原生安裝程式套件類型

應用程式套件必須是原生安裝程式。您無法在使用原生擴充功能的應用程式中，建立跨平台 AIR 套件 (一種 AIR 套件)，因為原生擴充功能通常包含原生程式碼。但是，通常原生擴充功能會支援相同 ActionScript API 的多個原生平台。在這種案例中，您可在不同的原生安裝程式套件中，使用相同的 ANE 檔案。

下列表格摘述使用 ADT 命令 -target 選項的值。

應用程式的目標平台	**-target**
Mac OS X 或 Windows 桌上型裝置	-target native -target bundle
Android	-target apk 或其他 Android 套件目標。
iOS	-target ipa-ad-hoc 或其他 iOS 套件目標
iOS 模擬器	-target ipa-test-interpreter-simulator -target ipa-debug-interpreter-simulator

指定擴充功能目錄

使用 ADT 選項 -extdir 來告知 ADT 包含原生擴充功能 (ANE 檔案) 的目錄。

如需此選項的相關資訊，請參閱第 155 頁 「檔案與路徑選項」。

請確定 ANE 檔案支援應用程式的目標裝置

提供 ANE 檔案時，原生擴充功能開發人員會通知您擴充功能支援哪些平台。您也可使用封存解壓縮工具來查看 ANE 檔案的內容。解壓縮的檔案包含每個支援平台的目錄。

封裝使用 ANE 檔案的應用程式時，瞭解哪些平台支援擴充功能，相當重要。請考慮下列規則：

- 若要建立 Android 應用程式套件，ANE 檔案必須包含 Android-ARM 平台。或者，ANE 檔案必須包含預設平台和至少一個其他平台。
- 若要建立 iOS 應用程式套件，ANE 檔案必須包含 iPhone-ARM 平台。或者，ANE 檔案必須包含預設平台和至少一個其他平台。
- 若要建立 iOS 模擬器應用程式套件，ANE 檔案必須包含 iPhone-x86 平台。
- 若要建立 MacOS X 應用程式套件，ANE 檔案必須包含 MacOS-x86 平台。或者，ANE 檔案必須包含預設平台和至少一個其他平台。
- 若要建立 Windows 應用程式套件，ANE 檔案必須包含 Windows-x86 平台。或者，ANE 檔案必須包含預設平台和至少一個其他平台。

第 10 章 ActionScript 編譯器

ActionScript 與 MXML 程式碼必須先經過編譯，才能包含在 AIR 應用程式中。若使用 Adobe Flash Builder 或 Adobe Flash Professional 之類的整合開發環境 (IDE)，IDE 會在幕後處理編譯事宜。不過，不使用 IDE 或使用建置程式碼時，您也可以從命令列叫用 ActionScript 編譯器以建立 SWF 檔案。

關於 Flex SDK 中的 AIR 命令列工具

您用來建立 Adobe AIR 應用程式的每個命令列工具都會呼叫對應的工具來建立應用程式：

- amxmlc 呼叫 mxmlc 來編譯應用程式類別
- acompc 呼叫 compc 來編譯元件庫與組件類別
- aasdoc 呼叫 asdoc 從原始碼註解產生文件檔案

Flex 與 AIR 版本的公用程式之間，唯一的差異是 AIR 版本會從 air-config.xml 檔案載入設定選項，而不是從 flex-config.xml 檔案。

如需有關 Flex SDK 工具及其命令列選項的詳細資訊，請參閱 Flex 文件。這裡所說明的 Flex SDK 工具不但提供您快速入門的基本說明，同時並指出建立 Flex 應用程式與建立 AIR 應用程式之間的差異。

更多說明主題

第 32 頁「使用 Flex SDK 建立您的第一個桌上型 AIR 應用程式」

編譯器設定

您通常會同時在命令列和一或多個設定檔中指定編譯選項。全域 Flex SDK 設定檔包含編譯器每次執行時使用的預設值。您可以依據自己的開發環境來編輯此檔案。Flex SDK 安裝的架構目錄中有兩個全域 Flex 設定檔。當您執行 amxmlc 編譯器時會使用 air-config.xml 檔案。此檔案藉由加入 AIR 元件庫來設定 AIR 專用的編譯器。當您執行 mxmlc 時會使用 flex-config.xml 檔案。

預設的設定值適用於瞭解 Flex 與 AIR 的運作狀況，但當您著手進行完整的專案時，請仔細檢查可用的選項。您可以在本機設定檔中提供編譯器選項的專案專用值，此值優先於指定專案的全域值。

備註：AIR 應用程式並無專用的編譯選項，但在編譯 AIR 應用程式時，您必須參考 AIR 元件庫。一般來說，您會在專案層級設定檔、建立工具 (例如 Ant) 檔案，或直接在命令列上參考這些元件庫。

編譯 AIR 的 MXML 和 ActionScript 原始檔案

您可以使用命令列 MXML 編譯器 (amxmlc) 來編譯 AIR 應用程式的 Adobe® ActionScript® 3.0 和 MXML 資源 (您不需要編譯 HTML 類型應用程式。若要在 Flash Professional 編譯 SWF，您只要將影片發佈至 SWF 檔即可)。

使用 amxmlc 時的基本命令列模式如下：

```
amxmlc [compiler options] -- MyAIRApp.mxml
```

其中 [compiler options] 可以指定用於編譯 AIR 應用程式的命令列選項。

amxmlc 命令會叫用標準的 Flex mxmlc 編譯器並搭配額外的參數 +configname=air。這個參數會指示編譯器使用 air-config.xml 檔案，而不是 flex-config.xml 檔案。使用 amxmlc 在其它方面都與使用 mxmlc 相同。

這個編譯器會載入 air-config.xml 設定檔，指定一般編譯 AIR 應用程式所需的 AIR 和 Flex 元件庫。您也可以使用本機的專案層級設定檔，覆寫全域設定的選項或新增其它選項。一般而言，建立本機設定檔最簡單的方式，就是編輯一份全域版本的副本。您可以搭配 -load-config 選項，載入本機檔案：

-load-config=project-config.xml 會覆寫全域選項。

-load-config+=project-config.xml 會在可以採用值以外項目的全域選項 (例如 -library-path 選項) 中加入其它值；只採用單一值的全域選項會遭到覆寫。

如果您為本機設定檔使用特殊的命名慣例，amxmlc 編譯器便會自動載入該本機檔案。舉例而言，如果主要 MXML 檔案為 RunningMan.mxml，則請將本機設定檔命名為 RunningMan-config.xml。現在，若要編譯應用程式，您只需輸入：

```
amxmlc RunningMan.mxml
```

RunningMan-config.xml 將會自動載入，因為它的檔名符合所編譯之 MXML 檔案的檔名。

amxmlc 範例

在下列範例會示範 amxmlc 編譯器的用法 (僅需要編譯應用程式的 ActionScript 和 MXML 資源)。

編譯 AIR MXML 檔案：

```
amxmlc myApp.mxml
```

編譯及設定輸出名稱：

```
amxmlc –output anApp.swf -- myApp.mxml
```

編譯 AIR ActionScript 檔案：

```
amxmlc myApp.as
```

指定編譯器設定檔：

```
amxmlc –load-config config.xml -- myApp.mxml
```

從其它設定檔新增額外選項：

```
amxmlc –load-config+=moreConfig.xml -- myApp.mxml
```

在命令列新增元件庫 (除了已經在設定檔中的元件庫以外)：

```
amxmlc –library-path+=/libs/libOne.swc,/libs/libTwo.swc  -- myApp.mxml
```

編譯 AIR MXML 檔案而不使用設定檔 (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
```

編譯 AIR MXML 檔案而不使用設定檔 (Mac OS X 或 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
```

編譯 AIR MXML 檔案以使用執行階段共享元件庫：

```
amxmlc -external-library-path+=../lib/myLib.swc -runtime-shared-libraries=myrsl.swf -- myApp.mxml
```

編譯 AIR MXML 檔案以使用 ANE (請務必針對 ANE 使用 -external-library-path)：

```
amxmlc -external-library-path+=../lib/myANE.ane -output=myAneApp.swf -- myAneApp.mxml
```

透過 Java 進行編譯 (其中類別路徑設定為包含 mxmlc.jar)：

```
java flex2.tools.Compiler +flexlib [Flex SDK 3]/frameworks +configname=air [additional compiler options] -
- myApp.mxml
```

flexlib 選項可用於識別 Flex SDK 架構目錄的位置，以供編譯器找到 flex_config.xml 檔案。

透過 Java 進行編譯 (不設定類別路徑)：

```
java -jar [Flex SDK 2]/lib/mxmlc.jar +flexlib [Flex SDK 3]/frameworks +configname=air [additional compiler
options] -- myApp.mxml
```

使用 Apache Ant 來叫用編譯器 (範例中使用 Java 工作來執行 mxmlc.jar)：

```
<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>
```

編譯 AIR 組件或程式庫 (Flex)

您可以使用組件編譯器 acompc 編譯 AIR 元件庫和獨立的組件。除了以下不同之處，acompc 組件編譯器的行為都與 amxmlc 編譯器相似：

- 您必須指定要將程式碼基底中的哪些類別納入元件庫或組件中。
- acompc 不會自動尋找本機設定檔。若要使用專案設定檔，您必須使用 –load-config 選項。

acompc 命令會叫用標準的 Flex compc 組件編譯器，但是會從 air-config.xml 檔案 (而非 flex-config.xml 檔案) 載入其設定選項。

組件編譯器設定檔

您可以使用本機設定檔，避免在命令列輸入 (而且可能是不正確的輸入) 來源路徑和類別名稱。將 -load-config 選項加入至 acompc 命令列，即可載入本機設定檔。

下列範例會示範用於建立具有 ParticleManager 和 Particle 類別之元件庫的設定，這兩個類別都在 com.adobe.samples.particles 套件中：類別檔案則位於 source/com/adobe/samples/particles 資料夾中。

```
<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>
```

若要使用名為 ParticleLib-config.xml 的設定檔來編譯元件庫，請輸入：

```
acompc -load-config ParticleLib-config.xml -output ParticleLib.swc
```

若要在命令列完整執行相同的命令，請輸入：

```
acompc -source-path source -include-classes com.adobe.samples.particles.Particle
com.adobe.samples.particles.ParticleManager -output ParticleLib.swc
```

(請在一行內輸入完整的命令，或為命令殼層使用行接續字元)。

acompc 範例

下列範例會假設您是使用名為 myLib-config.xml 的設定檔。

編譯 AIR 組件或元件庫：

```
acompc -load-config myLib-config.xml -output lib/myLib.swc
```

編譯執行階段共享元件庫：

```
acompc -load-config myLib-config.xml -directory -output lib
```

(請注意，在執行命令之前，lib 資料夾必須存在並且是空的)。

第 11 章 AIR Debug Launcher (ADL)

使用 AIR Debug Launcher (ADL)，即可在開發階段執行 SWF 類型和 HTML 類型的應用程式，也能在不需要先行封裝和安裝應用程式的情況下，直接執行應用程式。ADL 預設會使用 SDK 隨附的執行階段，這表示您不需要另外安裝執行階段，就可以使用 ADL。

ADL 會將追蹤陳述式和執行階段錯誤列印至標準輸出，但是不支援中斷點或其它除錯功能。您可以使用 Flash 除錯程式 (或是 Flash Builder 等整合開發環境) 來進行複雜問題的除錯工作。

備註：如果您的 trace() 陳述式未顯示在主控台上，請確定您尚未在 mm.cfg 檔案中指定 ErrorReportingEnable 或 TraceOutputFileEnable。如需有關這個檔案之平台特定位置的詳細資訊，請參閱編輯 mm.cfg 檔案。

AIR 支援直接進行除錯，因此您不需要執行階段的除錯版本 (使用 Adobe® Flash® Player 時則需要)。若要進行命令列除錯功能，您可以使用 Flash 除錯程式與 AIR Debug Launcher (ADL)。

Flash 除錯程式會散佈到 Flex SDK 目錄中。原生版本 (例如 Windows 上 fdb.exe) 則位於 bin 子目錄中。Java 版本位於 bin 子目錄中。AIR Debug Launcher (adl.exe) 位於 Flex SDK 安裝的 bin 目錄中 (並無獨立的 Java 版本)。

備註： fdb 會嘗試以 Flash Player 來啟動 AIR 應用程式，因此您無法直接使用 fdb 來啟動 AIR 應用程式。請改為讓 AIR 應用程式連線到執行中的 fdb 工作階段。

ADL 用法

如要使用 ADL 執行應用程式，請使用下列模式：

```
adl application.xml
```

其中 application.xml 是該應用程式的應用程式描述器檔案。

ADL 的完整語法為：

```
adl [-runtime runtime-directory]
    [-pubid publisher-id]
    [-nodebug]
    [-atlogin]
    [-profile profileName]
    [-screensize value]
    [-extdir extension-directory]
    application.xml
    [root-directory]
    [-- arguments]
```

(中括號 [] 中的項目為選擇性。)

-runtime runtime-directory：指定包含要使用之執行階段的目錄。如果未指定，則會使用與 ADL 程式相同之 SDK 中的執行階段目錄。如果您將 ADL 移至其 SDK 資料夾之外，請指定執行階段目錄。在 Windows 和 Linux 中，請指定其中含有 Adobe AIR 目錄的目錄 在 Mac OS X 中，請指定包含 Adobe AIR.framework 的目錄。

-pubid publisher-id：指定特定值做為這次執行 **AIR** 應用程式的發行者 **ID**。指定暫時性的發行者 ID 可以讓您測試 AIR 應用程式的功能 (例如經由本機連線來通訊)，這些功能會使用發行者 ID 協助唯一識別應用程式。至於 AIR 1.5.3，您也可以在應用程式描述器檔案中指定發行者 ID (而且不應該使用這個參數)。

備註：至於 AIR 1.5.3，不再自動計算發行者 ID 並指派給 AIR 應用程式。當您對現有的 AIR 應用程式建立更新程式時，可以指定發行者 ID，不過新的應用程式不需要，也不應該指定發行者 ID。

-nodebug：關閉除錯支援功能。如果有使用這項功能，應用程式程序便無法連接至 Flash 除錯程式，而且未處理例外的對話方塊便不會顯示 (不過，追蹤陳述式還是會列印至主控台視窗)。關閉除錯功能可以稍微提高應用程式的執行速度，也能更準確地模擬所安裝應用程式的執行模式。

-atlogin ：模擬登入時啟動應用程式。此旗標可讓您測試應用程式邏輯，查看此邏輯是否只有在將應用程式設為在使用者登入時啟動才會執行。使用 -atlogin 時，傳送到應用程式之 InvokeEvent 物件的 reason 屬性將為 login，而非 standard (除非應用程式已在執行中)。

-profile profileName：**ADL** 會使用指定的描述檔對應用程式進行除錯。profileName 可以是下列其中一個值：

- desktop
- extendedDesktop
- mobileDevice

如果應用程式描述器包含 supportedProfiles 元素，則使用 -profile 指定的描述檔必須是支援清單的成員。如果未使用 -profile 旗標，將使用應用程式描述器中第一個描述檔做為作用中描述檔。如果應用程式描述器未包含 supportedProfiles 元素且未使用 -profile 旗標，則使用 desktop 描述檔。

如需詳細資訊，請參閱 第 205 頁 「supportedProfiles」和第 211 頁 「裝置描述檔」。

-screensize 值 在桌上型中以 mobileDevice 描述檔執行應用程式時，所使用的模擬螢幕大小。請將螢幕大小指定為預先定義的螢幕類型，或指定成縱向版面的一般寬度和高度像素尺寸，以及全螢幕寬度和高度。若要依類型指定值，請使用下列其中一種預先定義的螢幕類型：

螢幕類型	一般寬度 x 高度	全螢幕寬度 x 高度
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
iPodRetina	640 x 920	640 x 960
iPod5Retina	640 x1096	640 x 1136
NexusOne	480 x 762	480 x 800
QVGA	240 x 320	240 x 320

螢幕類型	一般寬度 x 高度	全螢幕寬度 x 高度
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

若要直接指定螢幕像素尺寸，請使用下列格式：

```
widthXheight:fullscreenWidthXfullscreenHeight
```

請一律指定縱向版面的像素尺寸，也就是將寬度的值指定為小於高度的值。例如，NexusOne 螢幕可使用下列方式指定：

```
-screensize 480x762:480x800
```

-extdir extension-directory 執行階段應在其中搜尋原生擴充功能的目錄。目錄包含每個應用程式使用之原生擴充功能的子目錄每個子目錄都包含擴充功能的未封裝 ANE 檔案。例如：

```
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
```

使用 -extdir 參數時，請考量以下幾點：

- ADL 命令要求其中每指定目錄的副檔名都是 .ane。但是，字尾 ".ane" 前的檔案名稱可以是任何有效的檔案名稱。這不必與應用程式描述器檔案的 extensionID 元素一致。
- 您可指定 -extdir 參數一次以上。
- ADT 工具和 ADL 工具的 -extdir 參數用法不同。在 ADT 中，參數會指定包含 ANE 檔案的目錄。
- 您也可使用環境變數 AIR_EXTENSION_PATH 來指定擴充功能目錄。請參閱 第 160 頁 「ADT 環境變數」。

application.xml：應用程式描述器檔案。請參閱 第 173 頁 「AIR 應用程式描述器檔案」。應用程式描述器是 ADL 的唯一必要參數，且在大多數情況下是唯一需要的參數。

root-directory：指定要執行之應用程式的根目錄。如果未指定，則會使用包含該應用程式描述器檔案的目錄。

-- arguments 任何出現在 "--" 之後的字元字串都會傳遞至應用程式做為命令列引數。

備註：當您啟動已經在執行中的 AIR 應用程式時，並不會啟動該應用程式的新實體，而是會將 invoke 事件傳送至執行中的實體。

ADL 範例

在目前的目錄中執行應用程式：

```
adl myApp-app.xml
```

在目前目錄的子目錄中執行應用程式：

```
adl source/myApp-app.xml release
```

執行應用程式，並傳入兩個命令列引數「tick」和「tock」：

```
adl myApp-app.xml -- tick tock
```

使用特定執行階段執行應用程式：

```
adl -runtime /AIRSDK/runtime myApp-app.xml
```

在不支援除錯的情況下執行應用程式：

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

在行動裝置描述檔中執行應用程式，並模擬 Nexus One 螢幕大小：

```
adl -profile mobileDevice -screensize NexusOne myMobileApp-app.xml
```

使用 Apache Ant 執行應用程式 (範例中顯示的路徑是針對 Windows)：

```
<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 結束和錯誤代碼

下表說明 ADL 列印的結束代碼：

結束代碼	說明
0	順利啟動。ADL 在 AIR 應用程式結束之後結束。
1	成功叫用正在執行中的 AIR 應用程式。ADL 會立即結束。
2	用法錯誤。為 ADL 提供的引數不正確。
3	找不到執行階段。
4	無法啟動執行階段。通常，當應用程式中指定的版本與執行階段的版本不符時，就會發生這個錯誤。
5	發生未知原因的錯誤。

結束代碼	說明
6	找不到應用程式描述器檔案。
7	應用程式描述器的內容無效。這個錯誤通常表示 XML 的格式不正確。
8	找不到主要應用程式內容檔案 (這個檔案是在應用程式描述器檔案的 <content> 元素中指定)。
9	主要應用程式內容檔案並非有效的 SWF 或 HTML 檔案。
10	應用程式不支援以 -profile 選項指定的描述檔。
11	目前描述檔不支援 -screensize 引數。

第 12 章 AIR Developer Tool (ADT)

AIR Developer Tool (ADT) 是開發 AIR 應用程式的多用途命令列工具。您可以使用 ADT 來執行下列工作：

- 將 AIR 應用程式封裝為 .air 安裝檔案
- 將 AIR 應用程式封裝為原生安裝程式。例如，封裝成 Windows 上的 .exe 安裝程式檔案、iOS 上的 ipa，或是 Android 上的 .apk
- 將原生擴充功能封裝為 AIR 原生擴充功能 (ANE) 檔案
- 使用數位憑證簽署 AIR 應用程式
- 變更 (移轉) 用於應用程式更新的數位簽名
- 判斷連接到電腦的裝置
- 建立自我簽署的數位程式碼簽署憑證
- 在行動裝置上以遠端方式安裝、啟動和解除安裝應用程式
- 在行動裝置上以遠端方式安裝和解除安裝 AIR 執行階段

ADT 是包含在 AIR SDK 中的 Java 程式。您必須有 Java 1.5 或更新的版本才能使用 ADT。SDK 包括可叫用 ADT 的指令碼檔。若要使用此指令碼，Java 程式的位置必須包括在 Path 環境變數中。如果 AIR SDK bin 目錄也列在您的 Path 環境變數中，您便可以在命令列輸入 adt 並加上適當的引數來叫用 ADT (如果您不知道該如何設定 Path 環境變數，請參閱您的作業系統文件。如需進一步的協助，請至 第 263 頁 「Path 環境變數」取得在大部分電腦系統上設定 Path 的程序)。

至少需要 2GB 的電腦記憶體才能使用 ADT。如果您的記憶體少於此數量，ADT 執行時可能會發生記憶體不足的情況，特別是在封裝 iOS 的應用程式時。

假設 Java 與 AIR SDK bin 目錄都包括在 Path 變數中，您可以使用下列基本語法來執行 ADT：

```
adt -command options
```

備註：大多數的整合開發環境 (包括 Adobe Flash Builder 和 Adobe Flash Professional) 都會為您封裝並簽署 AIR 應用程式。當您已經使用這些開發環境時，通常不需要使用 ADT 來執行這些經常進行的工作。不過，您仍可能需要將 ADT 當作命令列工具來存取整合開發環境不支援的功能。此外，您可以在自動化建置程序中將 ADT 當成命令列工具使用。

ADT 命令

第一個傳遞到 ADT 的引數會指定下列其中一個命令。

- -package — 封裝 AIR 應用程式或 AIR 原生擴充功能 (ANE)。
- -prepare — 將 AIR 應用程式封裝為中繼檔案 (AIRI)，但不會進行簽署。您將無法安裝 AIRI 檔案。
- -sign — 簽署使用 -prepare 命令所產生的 AIRI 套件。-prepare 與 -sign 命令允許您在不同的時間執行封裝和簽署。您也可以使用 -sign 命令來簽署或重新簽署 ANE 套件。
- -migrate — 將移轉簽名套用至已簽署的 AIR 套件，這可讓您使用新的或更新的程式碼簽署憑證。
- -certificate — 建立自我簽署的數位程式碼簽署憑證。
- -checkstore — 確認是否可以存取金鑰儲存中的數位憑證。
- -installApp — 在裝置或裝置模擬器上安裝 AIR 應用程式。
- -launchApp — 在裝置或裝置模擬器上啟動 AIR 應用程式。
- -appVersion — 報告目前安裝在裝置或裝置模擬器上的 AIR 應用程式版本。

- -uninstallApp — 從裝置或裝置模擬器解除安裝 AIR 應用程式。
- -installRuntime — 在裝置或裝置模擬器上安裝 AIR 執行階段。
- -runtimeVersion — 報告目前安裝在裝置或裝置模擬器上的 AIR 執行階段版本。
- -uninstallRuntime — 從裝置或裝置模擬器解除安裝目前安裝的 AIR 執行階段。
- -version — 報告 ADT 版本號碼。
- -devices — 報告所連接之行動裝置或模擬器的裝置資訊。
- -help — 顯示命令與選項的清單。

許多 ADT 命令會共用相關的選項旗標組與參數。在此將分別詳細說明每個選項組：

- 第 153 頁「ADT 程式碼簽署選項」
- 第 155 頁「檔案與路徑選項」
- 第 156 頁「除錯程式連線選項」
- 第 156 頁「原生擴充功能選項」

ADT package 命令

-package 命令必須從主要應用程式目錄執行。此命令使用下列語法：

從組件應用程式檔案建立 AIR 套件：

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    -sampler
    -hideAneLibSymbols
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    FILE_OPTIONS
```

從組件應用程式檔案建立原生套件：

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    DEBUGGER_CONNECTION_OPTIONS
    -airDownloadURL URL
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    -platformsdk path
    FILE_OPTIONS
```

建立原生套件，其中包含來自組件應用程式檔案的原生擴充功能：

```
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
```

從 AIR 或 AIRI 檔案建立原生套件：

```
adt -package
    -target packageType
    NATIVE_SIGNING_OPTIONS
    output
    input_package
```

從元件原生擴充功能檔案中，建立原生擴充功能套件：

```
adt -package
    AIR_SIGNING_OPTIONS
    -target ane
    output
    ANE_OPTIONS
```

備註：您並未簽署 ANE 檔案，因此在本範例中 AIR_SIGNING_OPTIONS 參數是選擇性。

AIR_SIGNING_OPTIONS：AIR 簽署選項會識別用來簽署 AIR 安裝檔案的憑證。如需簽署選項的詳細資訊，請參閱 第 153 頁「ADT 程式碼簽署選項」。

-migrate：這個旗標會指定使用移轉憑證以及 AIR_SIGNING_OPTIONS 參數中指定的憑證來簽署應用程式。只有將桌上型應用程式封裝為原生安裝程式且應用程式使用原生擴充功能時，這個旗標才有效。在其他情況則會發生錯誤。移轉憑證的簽署選項會指定為 **MIGRATION_SIGNING_OPTIONS** 參數。如需有關那些簽署選項的詳細資訊，請參閱 第 153 頁「ADT 程式碼簽署選項」。使用 -migrate 旗標可讓您針對使用原生擴充功能的桌上型原生安裝程式應用程式建立更新，並且在原始憑證到期等情況中變更應用程式的程式碼簽署憑證。如需詳細資訊，請參閱第 169 頁「簽署 AIR 應用程式的更新版本」。

-package 命令的 -migrate 旗標可以在 AIR 3.6 及更新版本中取得。

-target 要建立的套件類型。支援的套件類型為：

- air — AIR 套件。“air” 是預設值，而且建立 AIR 或 AIRI 檔案時不需要指定 -target 旗標。
- airn — 延伸電視描述檔中裝置的原生應用程式套件。
- ane —AIR 原生擴充功能套件
- Android 套件目標：
  - apk — Android 套件。以此目標產生的套件只能安裝在 Android 裝置上，不能安裝在模擬器上。
  - apk-captive-runtime — 同時包含應用程式及固定 AIR 執行階段版本的 Android 套件。以此目標產生的套件只能安裝在 Android 裝置上，不能安裝在模擬器上。
  - apk-debug — 具有額外除錯資訊的 Android 套件 (應用程式中的 SWF 檔案也必須使用除錯支援來編譯)。
  - apk-emulator — 用於沒有除錯支援之模擬器的 Android 套件 (使用 apk-debug 目標以同時在模擬器與裝置上除錯)。
  - apk-profile — 支援應用程式效能與記憶體設定的 Android 套件。
- iOS 套件目標：
  - ipa-ad-hoc — 用於臨機操作散佈的 iOS 套件。
  - ipa-app-store — 用於 Apple App Store 散佈的 iOS 套件。
  - ipa-debug — 具有額外除錯資訊的 iOS 套件。(應用程式中的 SWF 檔案也必須使用除錯支援來編譯)。
  - ipa-test — 已編譯的 iOS 套件，但沒有最佳化或除錯資訊。
  - ipa-debug-interpreter — 功能等於除錯套件，但編譯速度更快。不過，ActionScript 著色器會進行解譯，而不是轉譯為機器碼。因此，解譯器套件中的程式碼執行比較慢。
  - ipa-debug-interpreter-simulator — 功能等於 ipa-debug-interpreter，但封裝成 iOS 模擬器。僅適用於 Macintosh。如果您使用這個選項，則也必須加入 -platformsdk 選項，以指定 iOS 模擬器 SDK 的路徑。

- ipa-test-interpreter — 功能等於測試套件，但編譯速度更快。不過，ActionScript 著色器會進行解譯，而不是轉譯為機器碼。因此，解譯器套件中的程式碼執行比較慢。
- ipa-test-interpreter-simulator — 功能等於 ipa-test-interpreter，但封裝成 iOS 模擬器。僅適用於 Macintosh。如果您使用這個選項，則也必須加入 -platformsdk 選項，以指定 iOS 模擬器 SDK 的路徑。

- native — 原生桌上型安裝程式。產生的檔案類型是執行命令之作業系統的原生安裝格式：
  - EXE — Windows
  - DMG — Mac
  - DEB — Ubuntu Linux (AIR 2.6 或更早版本)
  - RPM — Fedora 或 OpenSuse Linux (AIR 2.6 或更早版本)

  如需詳細資訊，請參閱第 49 頁 「封裝桌上型原生安裝程式」。

-sampler (僅限 iOS，AIR 3.4 及更新版本) 在 iOS 應用程式中啟用以遙測為基礎的 ActionScript 取樣程式。使用這個旗標可讓您以 Adobe Scout 來設定應用程式。雖然 Scout 可以設定任何 Flash 平台內容，但是啟用詳細的遙測可讓您深入了解 ActionScript 函數計時、DisplayList、Stage3D 顯示等等。請注意，使用這個旗標會對效能造成些微影響，因此，請勿用於實際執行的應用程式。

-hideAneLibSymbols (僅限 iOS，AIR 3.4 及更新版本) 應用程式開發人員可以使用多個來源中的多個原生擴充功能，而且如果 ANE 共用常見元件名稱，ADT 就會產生 「物件檔案中的元件重複」錯誤。在某些情況下，這個錯誤甚至會顯現其本身，如同執行階段發生當機般。您可以使用 hideAneLibSymbols 選項，指定只向 ANE 元件庫的來源顯示元件庫的元件 (yes) 還是全域顯示 (no)：

- **yes** — 隱藏 ANE 元件，以便解決非預期的元件衝突問題。
- **no** — (預設值) 不要隱藏 ANE 元件。這是 AIR 3.4 發行前版本的行為。

-embedBitcode (僅限 iOS、AIR 25 和更新版本) 應用程式開發人員可以指定 yes 或 no，藉此使用 embedBitcode 選項指定是否要在 iOS 應用程式中內嵌位元程式碼。若未指定，此參數的預設值為 no。

DEBUGGER_CONNECTION_OPTIONS 除錯程式連線選項會指定除錯套件是應該嘗試連線至另一部電腦上執行的遠端除錯程式，還是偵聽來自遠端除錯程式的連線。這組選項只支援行動除錯套件 (目標 apk-debug 和 ipa-debug)。如需有關這些選項的詳細資訊，請參閱第 156 頁 「除錯程式連線選項」。

-airDownloadURL 指定將 AIR 執行階段下載並安裝到 Android 裝置時所需的替代 URL。如果未指定，AIR 應用程式會將使用者重新導向至 Android Market 上的 AIR 執行階段 (如果使用者尚未安裝該執行階段)。

如果您的應用程式是透過其他市場散佈 (不是透過 Google 所經營的 Android Market)，您可能需要指定從該市場下載 AIR 執行階段時所需的 URL。某些替代市場不允許應用程式要求使用者從市場以外的位置進行下載。此選項只支援 Android 套件。

NATIVE_SIGNING_OPTIONS：原生簽署選項會識別用來簽署原生套件檔案的憑證。您可以使用這些簽署選項來套用原生作業系統 (而不是 AIR 執行階段) 所使用的簽名。這些選項與 AIR_SIGNING_OPTIONS 中的選項相同，您可以在第 153 頁 「ADT 程式碼簽署選項」中找到完整的說明。

Windows 與 Android 皆支援原生簽名。在 Windows 上，必須同時指定 AIR 簽署選項與原生簽署選項。但在 Android 上，只能指定原生簽署選項。

在許多情況下，您可以使用相同的程式碼簽署憑證來同時套用 AIR 與原生簽名。不過，這並非適用於所有的情況。例如，Google 針對將應用程式送出至 Android Market 的政策中，規定所有的應用程式都必須以有效期至少到 2033 年的憑證來簽署。這表示由知名的憑證授權單位所核發的憑證 (建議您在套用 AIR 簽名時使用的憑證) 不得用來簽署 Android 應用程式 (沒有任何憑證授權單位會核發有效期限如此長的程式碼簽署憑證)。

output：要建立的套件檔案名稱。您可以選擇是否要指定副檔名。如果不指定，系統便會新增適用於 -target 值和目前作業系統的副檔名。

app_descriptor：應用程式描述器檔案的路徑。指定路徑時，可以將路徑指定為相對於目前的目錄，或者指定為絕對路徑 (在 AIR 檔中，應用程式描述器檔案會重新命名為 application.xml)。

-platformsdk：目標裝置的平台 SDK 路徑：

- Android - AIR 2.6+ SDK 包括實作相關 ADT 命令所需的 Android SDK 工具。只有當您想要使用不同版本的 Android SDK 時，才需要設定此值。另外，如果已經設定 AIR_ANDROID_SDK_HOME 環境變數，便不需要在命令列上提供平台 SDK 路徑 (如果兩者均已設定，則會使用命令列上提供的路徑)。
- iOS - AIR SDK 隨附固定 iOS SDK。-platformsdk 選項可讓您使用外部 SDK 封裝應用程式，如此，就不會受限只能使用固定 iOS SDK。例如，如果您使用過最新的 iOS SDK 來建置擴充功能，則在封裝應用程式時可以指定該 SDK。此外，在搭配使用 ADT 與 iOS 模擬器時，您也必須一律加入 -platformsdk 選項，以指定 iOS 模擬器 SDK 的路徑。

-arch：應用程式開發人員可以使用這個引數來建立 x86 平台適用的 APK，它接受下列值：

- armv7 - 表示 Android armv7 平台的 ADT 套件 APK。
- x86 - 表示 Android x86 平台的 ADT 套件 APK。

未指定任何值時，armv7 即為預設值。

FILE_OPTIONS：指出要包括在套件中的應用程式檔案。如需有關檔案選項的詳細資訊，請參閱第 155 頁 「檔案與路徑選項」。從 AIR 或 AIRI 檔案建立原生套件時，請勿指定檔案選項。

input_airi：從 AIRI 檔案建立原生套件時指定。如果目標是 air (或未指定目標)，則需要使用 AIR_SIGNING_OPTIONS。

input_air：從 AIR 檔案建立原生套件時指定。請勿指定 AIR_SIGNING_OPTIONS。

ANE_OPTIONS 可識別建立原生擴充功能套件的選項和檔案。在第 156 頁 「原生擴充功能選項」中有詳述擴充功能套件選項。

ADT -package 命令範例

在目前目錄中，封裝 SWF 類型 AIR 應用程式的專屬應用程式檔案：

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf components.swc
```

在目前目錄中，封裝 HTML 類型 AIR 應用程式的專屬應用程式檔案：

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html AIRAliases.js image.gif
```

封裝目前工作目錄中的所有檔案和子目錄：

```
adt -package -storetype pkcs12 -keystore ../cert.p12 myApp.air myApp.xml .
```

備註：金鑰儲存檔案包含用於簽署應用程式的私有金鑰。因此，請務必不要將簽署憑證納入 AIR 套件中！如果您在 ADT 命令中使用萬用字元，請將金鑰儲存檔案放置於不同的位置，不要將它納入套件中。在上述範例中，金鑰儲存檔案 cert.p12 是位於上層目錄中。

僅封裝主要檔案和影像子目錄：

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf images
```

封裝 HTML 類型應用程式和 HTML 中的所有檔案、Script 和影像子目錄：

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml index.html AIRALiases.js html scripts
images
```

封裝位於工作目錄 (release/bin) 中的 application.xml 檔案和主要 SWF：

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml -C release/bin myApp.swf
```

從組建檔案系統中數個位置封裝資源。在下列範例中，應用程式資源在封裝之前位於下列資料夾中：

```
/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
```

從 /devRoot/myApp 目錄執行下列 ADT 命令：

```
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
```

產生下列封裝結構：

```
/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
```

針對簡單的 SWF 類型應用程式，將 ADT 當做 Java 程式執行 (不設定類別路徑)：

```
java -jar {AIRSDK}/lib/ADT.jar -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf
```

針對簡單的 HTML 類型應用程式，將 ADT 當做 Java 程式執行 (不設定類別路徑)：

```
java -jar {AIRSDK}/lib/ADT.jar -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html
AIRAliases.js
```

將 ADT 當做 Java 程式執行 (設定 Java 類別路徑以包含 ADT.jar 套件)：

```
java -com.adobe.air.ADT -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf
```

在 Apache Ant 中以 Java 工作的方式執行 ADT (通常最佳作法是直接在 Ant 指令碼中使用 ADT 命令)。範例中顯示的路徑適用於 Windows：

```
<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>
```

備註：在某些電腦系統上，可能會錯譯檔案系統路徑中的雙位元組字元。如果發生這個問題，請嘗試設定用來執行 ADT 的 JRE 以使用 UTF-8 字元集。而在 Mac 與 Linux 上啟動 ADT 的指令碼預設便已有此設定。在 Windows adt.bat 檔案中，或是當您從 Java 直接執行 ADT 時，請在 Java 命令列上指定 -Dfile.encoding=UTF-8 選項。

ADT prepare 命令

-prepare 命令會建立未簽署的 AIRI 套件。AIRI 套件無法獨立使用。使用 -sign 命令將 AIRI 檔案轉換為已簽署的 AIR 套件，或是使用 package 命令將 AIRI 檔案轉換為原生套件。

-prepare 命令使用下列語法：

```
adt -prepare output app_descriptor FILE_OPTIONS
```

output：建立的 AIRI 檔案名稱。

app_descriptor：應用程式描述器檔案的路徑。指定路徑時，可以將路徑指定為相對於目前的目錄，或者指定為絕對路徑 (在 AIR 檔中，應用程式描述器檔案會重新命名為 application.xml)。

FILE_OPTIONS：指出要包括在套件中的應用程式檔案。如需有關檔案選項的詳細資訊，請參閱第 155 頁 「檔案與路徑選項」。

ADT sign 命令

-sign 命令會簽署 AIRI 與 ANE 檔案。

-sign 命令使用下列語法：

```
adt -sign AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS：AIR 簽署選項會識別用來簽署套件檔案的憑證。如需有關簽署選項的詳細資訊，請參閱第 153 頁 「ADT 程式碼簽署選項」。

input ：要簽署的 AIRI 或 ANE 檔案名稱。

output：要建立的已簽署套件名稱。

如果 ANE 檔案已簽署，便會捨棄舊的簽名 (無法重新簽署 AIR 檔案 — 若要在應用程式更新使用新的簽名，請使用 -migrate 命令)。

ADT migrate 命令

-migrate 命令會將移轉簽名套用至 AIR 檔案。當您更新或變更數位憑證時，以及需要更新以舊憑證簽署的應用程式時，就必須使用移轉簽名。

如需有關以移轉簽名封裝 AIR 應用程式的詳細資訊，請參閱第 169 頁 「簽署 AIR 應用程式的更新版本」。

備註：您必須在憑證到期後算起的 365 天內套用移轉憑證。過了此寬限期後，您的應用程式更新將無法再以移轉簽名簽署。使用者可以先更新成以移轉簽名簽署的應用程式版本，然後再安裝最新的更新，或是他們可以解除安裝原始應用程式，然後安裝新的 AIR 套件。

若要使用移轉簽名，請先使用新的或更新的憑證來簽署您的 AIR 應用程式 (使用 -package 或 -sign 命令)，然後使用舊憑證與 -migrate 命令來套用移轉簽名。

-migrate 命令使用下列語法：

```
adt -migrate AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS：AIR 簽署選項會識別用來簽署 AIR 應用程式現有版本的原始憑證。如需簽署選項的詳細資訊，請參閱 第 153 頁 「ADT 程式碼簽署選項」。

input：已使用 「新」應用程式憑證來簽署的 AIR 檔案。

output：同時具有新憑證與舊憑證簽名的最終套件名稱。

用於輸出和輸入 AIR 檔案的檔案名稱必須不同。

備註：ADT migrate 命令無法與包含原生擴充功能的 AIR 桌上型應用程式一起搭配使用，因為那些應用程式是封裝為原生安裝程式，而非 .air 檔案。若要變更包含原生擴充功能之 AIR 桌上型應用程式的憑證，請搭配使用 第 143 頁 「ADT package 命令」與 -migrate 旗標來封裝應用程式。

ADT checkstore 命令

-checkstore 命令可讓您檢查金鑰儲存的有效性。命令使用下列語法：

```
adt -checkstore SIGNING_OPTIONS
```

SIGNING_OPTIONS：識別要驗證之金鑰儲存的簽署選項。如需簽署選項的詳細資訊，請參閱 第 153 頁 「ADT 程式碼簽署選項」。

ADT certificate 命令

-certificate 命令可讓您建立自我簽署數位程式碼簽署憑證。命令使用下列語法：

```
adt -certificate -cn name -ou orgUnit -o orgName -c country -validityPeriod years key-type output password
```

-cn ：指定做為新憑證一般名稱的字串。

-ou ：指定做為發行憑證的組織單位 (選擇性)。

-o ：指定做為發行憑證的組織 (選擇性)。

-c ：以兩個字母表示的 ISO-3166 國家 / 地區代碼。如果指定無效的代碼，便不會產生憑證 (選擇性)。

-validityPeriod：憑證有效的年數。如果未指定，則會指定五年的有效期 (選擇性)。

key_type：要對憑證使用的金鑰類型為 2048-RSA。

output：要產生之憑證檔案的路徑與檔案名稱。

password：存取新憑證的密碼。使用這個憑證簽署 AIR 檔時，需要提供這個密碼。

ADT installApp 命令

-installApp 命令會在裝置或模擬器上安裝應用程式。

您必須先解除安裝現有的應用程式，才能使用此命令重新安裝。

命令使用下列語法：

```
adt -installApp -platform platformName -platformsdk path-to-sdk -device deviceID -package fileName
```

-platform：裝置的平台名稱。指定 ios 或 android。

-platformsdk：目標裝置的平台 SDK 路徑 (選擇性)：

- Android - AIR 2.6+ SDK 包括實作相關 ADT 命令所需的 Android SDK 工具。只有當您想要使用不同版本的 Android SDK 時，才需要設定此值。另外，如果已經設定 AIR_ANDROID_SDK_HOME 環境變數，便不需要在命令列上提供平台 SDK 路徑 (如果兩者均已設定，則會使用命令列上提供的路徑)。
- iOS - AIR SDK 隨附固定 iOS SDK。-platformsdk 選項可讓您使用外部 SDK 封裝應用程式，如此，就不會受限只能使用固定 iOS SDK。例如，如果您使用過最新的 iOS SDK 來建置擴充功能，則在封裝應用程式時可以指定該 SDK。此外，在搭配使用 ADT 與 iOS 模擬器時，您也必須一律加入 -platformsdk 選項，以指定 iOS 模擬器 SDK 的路徑。

-device：指定連線裝置的 ios_simulator、序號 (Android) 或控制點 (iOS)。在 iOS 上，這個參數是必要的；在 Android 上，只有在一個以上的 Android 裝置或模擬器附加至電腦且執行時，才需要指定這個參數。如果未連接指定的裝置，ADT 會傳回結束代碼 14：裝置錯誤 (Android) 或指定的裝置無效 (iOS)。如果連接了多個裝置或模擬器，但未指定某一裝置，則 ADT 會傳回結束代碼 2：用法錯誤。

備註：若要直接將 IPA 檔案安裝到 iOS 裝置，可以在 AIR 3.4 中進行，並且需要 iTunes 10.5.0 及更新版本。

使用 adt -devices 命令 (在 AIR 3.4 及更新版本中提供使用) 可判斷連線裝置的控制點或序號。請注意，在 iOS 上使用的是控制點，而非裝置 UUID。如需詳細資訊，請參閱 第 153 頁「ADT 裝置命令」。

此外，在 Android 上，請使用 Android ADB 工具列出連接裝置的序號與執行中的模擬器：

```
adb devices
```

-package：要安裝的套件檔案名稱。在 iOS 上，這必須是 IPA 檔案。在 Android 上，這必須是 APK 套件。如果指定的套件已安裝，ADT 會傳回錯誤代碼 14：裝置錯誤。

ADT appVersion 命令

-appVersion 命令會報告在裝置或模擬器上安裝的應用程式版本。命令使用下列語法：

```
adt -appVersion -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform：裝置的平台名稱。指定 ios 或 android。

-platformsdk：目標裝置的平台 SDK 路徑：

- Android - AIR 2.6+ SDK 包括實作相關 ADT 命令所需的 Android SDK 工具。只有當您想要使用不同版本的 Android SDK 時，才需要設定此值。另外，如果已經設定 AIR_ANDROID_SDK_HOME 環境變數，便不需要在命令列上提供平台 SDK 路徑 (如果兩者均已設定，則會使用命令列上提供的路徑)。
- iOS - AIR SDK 隨附固定 iOS SDK。-platformsdk 選項可讓您使用外部 SDK 封裝應用程式，如此，就不會受限只能使用固定 iOS SDK。例如，如果您使用過最新的 iOS SDK 來建置擴充功能，則在封裝應用程式時可以指定該 SDK。此外，在搭配使用 ADT 與 iOS 模擬器時，您也必須一律加入 -platformsdk 選項，以指定 iOS 模擬器 SDK 的路徑。

-device：指定 ios_simulator 或裝置的序號。只有在電腦連接和執行多個 Android 裝置或模擬器時，才需要指定裝置。如果未連接指定的裝置，ADT 會傳回結束代碼 14：裝置錯誤。如果連接了多個裝置或模擬器，但未指定某一裝置，則 ADT 會傳回結束代碼 2：用法錯誤。

在 Android 上，請使用 Android ADB 工具列出連接裝置的序號與執行中的模擬器：

```
adb devices
```

-appid：安裝之應用程式的 AIR 應用程式 ID。如果裝置上沒有安裝指定 ID 的應用程式，則 ADT 會傳回結束代碼 14：裝置錯誤。

ADT launchApp 命令

-launchApp 命令會在裝置或模擬器上執行安裝的應用程式。命令使用下列語法：

```
adt -launchApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform：裝置的平台名稱。指定 ios 或 android。

-platformsdk：目標裝置的平台 SDK 路徑：

- Android - AIR 2.6+ SDK 包括實作相關 ADT 命令所需的 Android SDK 工具。只有當您想要使用不同版本的 Android SDK 時，才需要設定此值。另外，如果已經設定 AIR_ANDROID_SDK_HOME 環境變數，便不需要在命令列上提供平台 SDK 路徑 (如果兩者均已設定，則會使用命令列上提供的路徑)。
- iOS - AIR SDK 隨附固定 iOS SDK。-platformsdk 選項可讓您使用外部 SDK 封裝應用程式，如此，就不會受限只能使用固定 iOS SDK。例如，如果您使用過最新的 iOS SDK 來建置擴充功能，則在封裝應用程式時可以指定該 SDK。此外，在搭配使用 ADT 與 iOS 模擬器時，您也必須一律加入 -platformsdk 選項，以指定 iOS 模擬器 SDK 的路徑。

-device：指定 ios_simulator 或裝置的序號。只有在電腦連接和執行多個 Android 裝置或模擬器時，才需要指定裝置。如果未連接指定的裝置，ADT 會傳回結束代碼 14：裝置錯誤。如果連接了多個裝置或模擬器，但未指定某一裝置，則 ADT 會傳回結束代碼 2：用法錯誤。

在 Android 上，請使用 Android ADB 工具列出連接裝置的序號與執行中的模擬器：

```
adb devices
```

-appid：安裝之應用程式的 AIR 應用程式 ID。如果裝置上沒有安裝指定 ID 的應用程式，則 ADT 會傳回結束代碼 14：裝置錯誤。

ADT uninstallApp 命令

-uninstallApp 命令會在遠端裝置或模擬器上完全移除已安裝的應用程式。命令使用下列語法：

```
adt -uninstallApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid applicationID
```

-platform：裝置的平台名稱。指定 ios 或 android。

-platformsdk：目標裝置的平台 SDK 路徑：

- Android - AIR 2.6+ SDK 包括實作相關 ADT 命令所需的 Android SDK 工具。只有當您想要使用不同版本的 Android SDK 時，才需要設定此值。另外，如果已經設定 AIR_ANDROID_SDK_HOME 環境變數，便不需要在命令列上提供平台 SDK 路徑 (如果兩者均已設定，則會使用命令列上提供的路徑)。
- iOS - AIR SDK 隨附固定 iOS SDK。-platformsdk 選項可讓您使用外部 SDK 封裝應用程式，如此，就不會受限只能使用固定 iOS SDK。例如，如果您使用過最新的 iOS SDK 來建置擴充功能，則在封裝應用程式時可以指定該 SDK。此外，在搭配使用 ADT 與 iOS 模擬器時，您也必須一律加入 -platformsdk 選項，以指定 iOS 模擬器 SDK 的路徑。

-device：指定 ios_simulator 或裝置的序號。只有在電腦連接和執行多個 Android 裝置或模擬器時，才需要指定裝置。如果未連接指定的裝置，ADT 會傳回結束代碼 14：裝置錯誤。如果連接了多個裝置或模擬器，但未指定某一裝置，則 ADT 會傳回結束代碼 2：用法錯誤。

在 Android 上，請使用 Android ADB 工具列出連接裝置的序號與執行中的模擬器：

```
adb devices
```

-appid：安裝之應用程式的 AIR 應用程式 ID。如果裝置上沒有安裝指定 ID 的應用程式，則 ADT 會傳回結束代碼 14：裝置錯誤。

ADT installRuntime 命令

-installRuntime 命令會在裝置上安裝 AIR 執行階段。

您必須先解除安裝現有版本的 AIR 執行階段，才能使用此命令重新安裝。

命令使用下列語法：

```
adt -installRuntime -platform platformName -platformsdk path_to_sdk -device deviceID -package fileName
```

-platform：裝置的平台名稱。目前只支援在 Android 平台使用此命令。請使用 android 這個名稱。

-platformsdk：目標裝置的平台 SDK 路徑。目前唯一支援的平台 SDK 是 Android。AIR 2.6+ SDK 包括實作相關 ADT 命令所需的 Android SDK 工具。只有當您想要使用不同版本的 Android SDK 時，才需要設定此值。另外，如果已經設定 AIR_ANDROID_SDK_HOME 環境變數，便不需要在命令列上提供平台 SDK 路徑 (如果兩者均已設定，則會使用命令列上提供的路徑)。

-device：裝置的序號。只有在電腦連接和執行多個裝置或模擬器時，才需要指定裝置。如果未連接指定的裝置，ADT 會傳回結束代碼 14：裝置錯誤。如果連接了多個裝置或模擬器，但未指定某一裝置，則 ADT 會傳回結束代碼 2：用法錯誤。

在 Android 上，請使用 Android ADB 工具列出連接裝置的序號與執行中的模擬器：

```
adb devices
```

-package：要安裝的執行階段檔案名稱。在 Android 上，這必須是 APK 套件。如果未指定任何套件，便會從 AIR SDK 中為裝置或模擬器選擇適當的執行階段。如果已經安裝執行階段，ADT 會傳回錯誤代碼 14：裝置錯誤。

ADT runtimeVersion 命令

-runtimeVersion 命令會報告在裝置或模擬器上安裝的 AIR 執行階段版本。命令使用下列語法：

```
adt -runtimeVersion -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform：裝置的平台名稱。目前只支援在 Android 平台使用此命令。請使用 android 這個名稱。

-platformsdk：目標裝置的平台 SDK 路徑。目前唯一支援的平台 SDK 是 Android。AIR 2.6+ SDK 包括實作相關 ADT 命令所需的 Android SDK 工具。只有當您想要使用不同版本的 Android SDK 時，才需要設定此值。另外，如果已經設定 AIR_ANDROID_SDK_HOME 環境變數，便不需要在命令列上提供平台 SDK 路徑 (如果兩者均已設定，則會使用命令列上提供的路徑)。

-device：裝置的序號。只有在電腦連接和執行多個裝置或模擬器時，才需要指定裝置。如果未安裝執行階段或未連接指定的裝置，ADT 會傳回結束代碼 14：裝置錯誤。如果連接了多個裝置或模擬器，但未指定某一裝置，則 ADT 會傳回結束代碼 2：用法錯誤。

在 Android 上，請使用 Android ADB 工具列出連接裝置的序號與執行中的模擬器：

```
adb devices
```

ADT uninstallRuntime 命令

-uninstallRuntime 命令會從裝置或模擬器完全移除 AIR 執行階段。命令使用下列語法：

```
adt -uninstallRuntime -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform：裝置的平台名稱。目前只支援在 Android 平台使用此命令。請使用 android 這個名稱。

-platformsdk：目標裝置的平台 SDK 路徑。目前唯一支援的平台 SDK 是 Android。AIR 2.6+ SDK 包括實作相關 ADT 命令所需的 Android SDK 工具。只有當您想要使用不同版本的 Android SDK 時，才需要設定此值。另外，如果已經設定 AIR_ANDROID_SDK_HOME 環境變數，便不需要在命令列上提供平台 SDK 路徑 (如果兩者均已設定，則會使用命令列上提供的路徑)。

-device：裝置的序號。只有在電腦連接和執行多個裝置或模擬器時，才需要指定裝置。如果未連接指定的裝置，ADT 會傳回結束代碼 14：裝置錯誤。如果連接了多個裝置或模擬器，但未指定某一裝置，則 ADT 會傳回結束代碼 2：用法錯誤。

在 Android 上，請使用 Android ADB 工具列出連接裝置的序號與執行中的模擬器：

```
adb devices
```

ADT 裝置命令

ADT -help 命令會顯示目前所連接之行動裝置和模擬器的裝置 ID：

```
adt -devices -platform iOS|android
```

-platform：要檢查的平台名稱。請指定 android 或 iOS。

備註：在 iOS 上，這個命令需要 iTunes 10.5.0 及更新版本。

ADT help 命令

ADT -help 命令會顯示命令列選項的簡短說明：

```
adt -help
```

help 輸出會使用下列符號慣例：

- <> — 位於角括號內的項目為您必須提供的資訊。
- () — 位於括號內的項目為在 help 命令輸出中被視為群組的選項。
- ALL_CAPS — 以大寫字母拼出的項目為會另外說明的選項組。
- | — 或。例如，(A | B) 表示項目 A 或項目 B。
- ? — 0 或 1。項目後面的問號表示項目是選擇性項目，而且使用時只能執行一個實體。
- * — 0 或更多。項目後面的星號表示項目是選擇性項目，而且可以執行任何數目的實體。
- + — 1 或更多。項目後面的加號表示項目是必要項目，而且可以執行多個實體。
- 沒有符號 — 如果項目沒有字尾符號，則表示該項目為必要項目，而且只能執行一個實體。

ADT 選項組

數個 ADT 命令會共用同一組常用選項。

ADT 程式碼簽署選項

ADT 會使用 Java 密碼架構 (JCA) 存取用於簽署 AIR 應用程式的私密金鑰和憑證。這些簽署選項會識別金鑰儲存，以及金鑰儲存中的私密金鑰和憑證。

金鑰儲存必須包含私密金鑰和相關的憑證鏈結。如果簽署憑證鏈結至電腦上受信任的憑證，則憑證的一般名稱欄位內容就會以發行者名稱的形式顯示在 AIR 安裝對話方塊中。

ADT 會要求憑證符合 x509v3 標準 (RFC3280)，並且包含具有適當值的 「擴充金鑰使用方式」擴充功能供程式碼簽署使用。憑證中定義的限制都將予以遵守，且這些限制可能會禁止使用某些憑證來簽署 AIR 應用程式。

備註：ADT 會適時使用 Java 執行階段環境 Proxy 設定連線至網際網路資源，以檢查憑證撤銷清單並取得時間戳記。如果您在使用 ADT 連線至這些網際網路資源時遇到問題，而且網路需要特定的 Proxy 設定，則您可能必須設定 JRE Proxy 設定。

AIR 簽署選項語法

簽署選項使用下列語法 (在單一命令列上)：

```
-alias aliasName
-storetype type
-keystore path
-storepass password1
-keypass password2
-providerName className
-tsa url
```

-alias：金鑰儲存中金鑰的別名。如果金鑰儲存中只包含一個憑證，就不需要指定別名。如果未指定別名，ADT 就會使用金鑰儲存中的第一個金鑰。

並非所有的金鑰儲存管理應用程式都允許將別名指定給憑證。例如，使用 Windows 系統金鑰存放區時，請使用憑證的辨別名稱做為別名。您可以使用 Java Keytool 公用程式列出可用的憑證，如此即可決定別名。例如，您可以執行下列命令：

```
keytool -list -storetype Windows-MY
```

便會產生憑證的下列輸出：

```
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
```

若要在 ADT 命令列參考這個憑證，請將別名設定為：

```
CN=TestingCert,OU=QE,O=Adobe,C=US
```

在 Mac OS X 中，「鑰匙圈」內憑證的別名就是 「鑰匙圈存取」應用程式中顯示的名稱。

-storetype ：金鑰儲存的類型，由金鑰儲存實作決定。大部分情況下，在安裝 Java 時一併安裝的預設金鑰儲存實作可支援 JKS 和 PKCS12 類型。Java 5.0 包含對 PKCS11 類型的支援 (可用於存取硬體字符上的金鑰儲存) 和對 Keychain 類型的支援 (可用於存取 Mac OS X 鑰匙圈)。Java 6.0 包含對 MSCAPI 類型 (適用於 Windows) 的支援。如果已經安裝並設定其它 JCA 提供者，那麼也可以使用其它金鑰儲存類型。如果未指定金鑰儲存類型，則會使用預設 JCA 提供者的預設類型。

儲存類型	金鑰儲存格式	最小 **Java** 版本
JKS	Java 金鑰儲存檔 (.keystore)	1.2
PKCS12	PKCS12 檔 (.p12 或 .pfx)	1.4
PKCS11	硬體字符	1.5
KeychainStore	Mac OS X 鑰匙圈	1.5
Windows-MY 或 Windows-ROOT	MSCAPI	1.6

-keystore ：檔案架構儲存類型的金鑰儲存檔案路徑。

-storepass ：存取金鑰儲存所需的密碼。如果未指定，ADT 就會提示您輸入密碼。

-keypass ：存取用來簽署 AIR 應用程式之私密金鑰時所需的密碼。如果未指定，ADT 就會提示您輸入密碼。

備註：如果您將密碼輸入做為 ADT 命令的一部分，則密碼字元便會儲存在命令列記錄中。因此，當憑證的安全性很重要時，不建議您使用 -keypass 或 -storepass 選項。另外請注意，當您省略密碼選項時，便不會顯示在密碼提示時所輸入的字元 (基於相同的安全性考量)。只要輸入密碼並按 Enter 鍵即可。

-providerName ：指定之金鑰儲存類型的 JCA 提供者。如果未指定，ADT 就會根據該類型的金鑰儲存，使用預設的提供者。

-tsa ：指定符合 RFC3161 之時間戳記伺服器的 URL，以便為數位簽名加上時間戳記。如果未指定 URL，則會使用 Geotrust 提供的預設時間戳記伺服器。AIR 應用程式的簽名在加上時間戳記之後，即使簽署憑證過期，仍然可以安裝應用程式，因為時間戳記已經確認當初簽署的憑證有效。

如果 ADT 無法連接至時間戳記伺服器，簽署作業就會取消，並且不會產生套件。指定 -tsa none 即可停用時間戳記功能。但是，如果 AIR 應用程式在封裝時沒有加上時間戳記，則簽署憑證過期之後就無法安裝應用程式。

備註：許多的簽署選項與 Java Keytool 公用程式的選項相同。您可以使用 Keytool 公用程式，檢查及管理 Windows 中的金鑰儲存。在 Mac OS X 中，您也可以使用 Apple® 安全性公用程式來達到上述目的。

-provisioning-profile：Apple iOS 佈建檔案 (僅封裝 iOS 應用程式時需要)。

簽署選項範例
使用 .p12 檔案進行簽署：

```
-storetype pkcs12 -keystore cert.p12
```

使用預設的 Java 金鑰儲存進行簽署：

```
-alias AIRcert -storetype jks
```

使用特定的 Java 金鑰儲存進行簽署：

```
-alias AIRcert -storetype jks -keystore certStore.keystore
```

使用 Mac OS X 鑰匙圈進行簽署：

```
-alias AIRcert -storetype KeychainStore -providerName Apple
```

使用 Windows 系統金鑰儲存進行簽署：

```
-alias cn=AIRCert -storeype Windows-MY
```

使用硬體字符進行簽署 (請參考字符廠商關於設定 Java 的指示以使用字符及提供正確的 providerName 值)：

```
-alias AIRCert -storetype pkcs11 -providerName tokenProviderName
```

在沒有內嵌時間戳記的情況下進行簽署：

```
-storetype pkcs12 -keystore cert.p12 -tsa none
```

檔案與路徑選項

檔案與路徑選項會指定要包括在套件中的所有檔案。檔案與路徑選項使用下列語法：

```
files_and_dirs -C dir files_and_dirs -e file_or_dir dir -extdir dir
```

files_and_dirs ：要封裝在 AIR 檔案中的檔案和目錄。您可以指定任何檔案和目錄，並以空白字元加以區隔。如果您列出目錄，則除了隱藏檔以外，其中所有的檔案和子目錄都會加入至套件中 (此外，如果已經直接指定應用程式描述器檔案，或透過萬用字元或目錄延伸字元指定應用程式描述器檔案，則這個檔案會遭忽略，並且不會再次加入至套件中)。指定的檔案和目錄都必須位於目前目錄或其中的一個子目錄內。使用 -C 選項即可變更目前的目錄。

重要事項：當 –C 選項後面接著 file_or_dir 引數時，便不可以在這些引數中使用萬用字元 (命令列殼層會在將這些引數傳遞至 ADT 之前延伸萬用字元，如此會造成 ADT 在錯誤的位置搜尋檔案)。不過，您還是可以使用點字元「.」來表示目前的目錄。例如：-C assets . 會將 assets 目錄中的所有項目 (包括所有子目錄) 複製到應用程式套件的根層級。

-C dir files_and_dirs：在處理已加入應用程式套件的後續檔案和目錄 (**files_and_dirs** 中所指定) 之前，將工作目錄變更為 dir 的值。所有檔案或目錄都會加入至應用程式套件的根層級。–C 選項的使用次數沒有限制，只要能從檔案系統中各處加入檔案即可。如果為 dir 指定相對路徑，則在解析路徑時，一定會從原始的工作目錄進行解析。

當 ADT 處理已納入套件中的檔案和目錄時，便會儲存目前目錄和目標檔案之間的相對路徑。安裝套件時，這些路徑就會延伸至應用程式目錄結構內。因此，指定 -C release/bin lib/feature.swf 就會將 release/bin/lib/feature.swf 檔放置於根應用程式資料夾的 lib 子目錄中。

-e file_or_dir dir ：將檔案或目錄放入指定的套件目錄。封裝 ANE 檔案時，無法使用此選項。

備註：應用程式描述器檔案的 <content> 元素必須指定應用程式套件目錄樹狀結構中主要應用程式檔案的最終位置。

-extdir dir*dir* 值是搜尋原生擴充功能 (ANE 檔案) 的目錄名稱。指定目前目錄的絕對路徑，或相對路徑。您可多次指定 -extdir 選項。

指定的目錄包含應用程式使用之原生擴充功能的 ANE 檔案。此目錄中每一個 ANE 檔案都有副檔名 .ane。不過，.ane 副檔名前面的檔案名稱「不」一定要符合應用程式描述器檔案的 extensionID 元素值。

例如，如果您使用 -extdir ./extensions，目錄 extensions 可看成：

```
extensions/
    extension1.ane
    extension2.ane
```

備註：ADT 工具和 ADL 工具的 -extdir 選項用法不同。在 ADL 中，選項會指定包含子目錄的目錄，每一個目錄都包含未封裝的 ANE 檔案。在 ADT 中，選項會指定包含 ANE 檔案的目錄。

除錯程式連線選項

當套件的目標為 apk-debug、ipa-debug 或 ipa-debug-interpreter 時，您可以使用連線選項，指定應用程式嘗試連線至遠端除錯程式 (通常用於 wifi 除錯)，或是偵聽來自遠端除錯程式的傳入連線 (通常用於 USB 除錯)。使用 -connect 選項可連線至除錯程式；使用 -listen 選項則可接受來自除錯程式透過 USB 連線的連線。這些選項互相排斥；也就是說，您無法同時使用。

-connect 選項使用下列語法：

```
-connect hostString
```

-connect：如果指定，應用程式將嘗試連線至遠端除錯程式。

hostString：識別執行 Flash 除錯工具 FDB 之電腦的字串。若未指定，應用程式將嘗試連線於建立套件的電腦上執行的除錯程式。主機字串可以是完整的電腦網域名稱：machinename.subgroup.example.com，或是 IP 位址：192.168.4.122。如果找不到指定 (或預設) 機器，則執行階段將顯示一個要求有效主機名稱的對話方塊。

-listen 選項使用下列語法：

```
-listen port
```

-listen 如果存在，則執行階段會等待來自遠端除錯程式的連線。

port (選擇性) 要偵聽的連接埠。根據預設，執行階段會偵聽連接埠 7936。如需有關使用 -listen 選項的詳細資訊，請參閱第 91 頁「透過 USB 使用 FDB 遠端除錯」。

Android 應用程式設定選項

套件的目標為 apk-profile 時，可使用設定選項來指定要用於效能和記憶體設定的預先載入 SWF 檔案。設定選項使用下列語法：

```
-preloadSWFPath directory
```

-preloadSWFPath：如果指定，應用程式將嘗試在指定的目錄尋找預先載入的 SWF。若未指定，ADT 將加入 AIR SDK 中的預先載入 SWF 檔案。

directory：包含設定預先載入 SWF 檔案的目錄。

原生擴充功能選項

原生擴充功能選項會指定封裝原生擴充功能的 ANE 檔案時使用的選項和檔案。在 -target 選項為 ane 的 ADT package 命令中，使用這些選項。

```
extension-descriptor -swc swcPath
   -platform platformName
  -platformoptions path/platform.xml
   FILE_OPTIONS
```

extension-descriptor：原生擴充功能的描述器檔案。

-swc SWC 檔案，包含原生擴充功能的 ActionScript 程式碼和資源。

-platform：此 ANE 檔案支援的平台名稱。您可包含多個 -platform 選項，每一都有專屬的 FILE_OPTIONS。

-platformoptions 平台選項 (platform.xml) 檔案的路徑。使用這個檔案可以指定擴充功能所使用的非預設連結器選項、共用元件庫和協力廠商靜態元件庫。如需詳細資訊和範例，請參閱 iOS 原生元件庫。

FILE_OPTIONS 會識別要加入套件中的原生平台檔案，例如要加入原生擴充功能套件的靜態元件庫。如需檔案選項的詳細資訊，請參閱第 155 頁 「檔案與路徑選項」。(請注意，封裝 ANE 檔案時不能使用 -e 選項)。

更多說明主題

封裝原生擴充功能

ADT 錯誤訊息

下表列出 ADT 程式可能回報的錯誤以及可能的發生原因：

應用程式描述器驗證錯誤

錯誤代碼	說明	備註
100	無法剖析應用程式描述器	請檢查應用程式描述器是否有 XML 語法錯誤，例如，未封閉的標籤。
101	遺失命名空間	請將遺失的命名空間加入。
102	無效的命名空間	請檢查命名空間的拼字。
103	未預期的元素或特質	請移除違規的元素和特質。描述器檔案中並不允許使用自訂值。 請檢查元素名稱和特質名稱的拼字。 請確認各元素是否放在正確的父元素中、各特質是否搭配正確的元素使用。
104	遺失元素或特質	請加入所需的元素或特質。
105	元素或特質包含無效的值	請對違規的值進行修正。
106	不合法的視窗特質組合	某些視窗設定並不能相互搭配使用，例如 transparency = true 和 systemChrome = standard。請變更其中一個不相容的設定。
107	視窗的最小尺寸超出視窗的最大尺寸	請對最大尺寸或最小尺寸的設定進行變更。
108	已在之前的元素中使用特質	
109	重複的元素。	請移除重複的元素。
110	至少需要一個指定類型的元素。	請新增遺失的元素。
111	應用程式描述器中列出的描述檔均不支援原生擴充功能。	將描述檔新增至支援原生擴充功能的 supportedProfies 清單中。
112	AIR 目標不支援原生擴充功能。	選擇支援原生擴充功能的目標。
113	必須同時提供 <nativeLibrary> 與 <initializer>。	必須為原生擴充功能中的每個原生元件庫指定初始設定式函數。
114	找到 <finalizer>，但沒有 <nativeLibrary>。	除非平台使用原生元件庫，否則請勿指定結束設定式。

錯誤代碼	說明	備註
115	預設平台不能包含原生實作。	請勿在預設平台元素中指定原生元件庫。
116	不支援此目標的瀏覽器引動過程。	指定封裝目標的 <allowBrowserInvocation> 元素不可為 true。
117	此目標必須至少有一個命名空間 n 來封裝原生擴充功能。	將應用程式描述器中的 AIR 命名空間變更為支援的值。

如需有關命名空間、元素、特質及其有效值的相關資訊，請參閱 第 173 頁 「AIR 應用程式描述器檔案」。

應用程式圖示錯誤

錯誤代碼	說明	備註
200	無法開啟圖示檔	請檢查檔案是否存在於指定的路徑內。 請使用其它的應用程式來確認檔案是否能夠開啟。
201	圖示的尺寸有誤	圖示的大小 (以像素為單位) 必須符合 XML 標籤。例如，假設應用程式描述器元素為： <image32x32>icon.png</image32x32> icon.png 中的影像必須剛好 32x32 像素。
202	圖示檔含有未受支援的影像格式	只有 PNG 格式受到支援。請在封裝應用程式之前先為其它格式的影像進行轉換。

應用程式檔案錯誤

錯誤代碼	說明	備註
300	遺失檔案或無法開啟檔案	找不到命令列所指定的檔案，或無法開啟該檔案。
301	遺失應用程式描述器檔案或無法開啟	無法在指定的路徑中找到應用程式描述器檔案，或無法開啟該檔案。
302	套件中未含根內容檔案	應用程式描述器在 <content> 元素中所參照的 SWF 或 HTML 檔案都必須加入套件中，只要將其納入 ADT 命令列的檔案清單即可。
303	套件中未含圖示檔	應用程式描述器內指定的圖示檔都必須加入套件中，只要將其納入 ADT 命令列的檔案清單即可。圖示檔並不會自動加入。
304	初始視窗內容無效	應用程式描述器在 <content> 元素中所參照的檔案未被視為有效的 HTML 或 SWF 檔案。
305	初始視窗內容的 SWF 版本超出命名空間版本的支援範圍	應用程式描述器在 <content> 元素中參照的 SWF 版本檔案並未受到描述器命名空間內指定的 AIR 版本所支援。例如，若嘗試封裝 SWF10 (Flash Player 10) 檔案來做為 AIR 1.1 應用程式的初始內容，就會產生這個錯誤。
306	不支援描述檔。	不支援您在應用程式描述器檔案中指定的描述檔。請參閱 第 205 頁 「supportedProfiles」。
307	命名空間必須至少為 nnn。	針對應用程式中使用的功能使用適當的命名空間 (例如 2.0 命名空間)。

其它錯誤的結束代碼

結束代碼	說明	備註
2	用法錯誤	檢查命令列引數是否有誤
5	未知的錯誤	此錯誤表示所發生的情形無法以一般的錯誤狀況來說明。可能的根本原因包括 ADT 與 Java Runtime Environment 不相容、ADT 或 JRE 的安裝毀損，以及 ADT 的程式設計有誤。
6	無法寫至輸出目錄	請確認所指定的 (或隱含的) 輸出目錄是否可提供存取，以及負責儲存的磁碟機是否具有足夠的空間。
7	無法存取憑證	請確認金鑰儲存的路徑是否已正確指定， 並檢查金鑰儲存內的憑證是否可供存取。Java 1.6 Keytool 公用程式可用來協助進行憑證存取問題的疑難排解。
8	無效的憑證	憑證檔案的格式有誤、遭到竄改、已過期或撤銷。
9	無法簽署 AIR 檔	請針對傳至 ADT 的簽署選項進行檢查。
10	無法建立時間戳記	ADT 無法連線至時間戳記伺服器。如果您透過 Proxy 伺服器連線至網際網路，就可能需要調整 JRE Proxy 設定。
11	憑證建立錯誤	請對用來建立簽名的命令列引數進行檢查。
12	無效的輸入	請對檔案路徑以及其它在命令列上傳遞至 ADT 的引數進行檢查。
13	遺失裝置 SDK	請確認裝置 SDK 組態。ADT 找不到執行指定命令所需的裝置 SDK。
14	裝置錯誤	ADT 因為裝置限制或裝置問題而無法執行命令。例如，當您嘗試解除安裝未實際安裝的應用程式時，便會發出此結束代碼。
15	沒有裝置	請確認已連接並開啟裝置，或有正在執行的模擬器。
16	遺失 GPL 組件	目前的 AIR SDK 中沒有執行要求作業所需的全部組件。
17	裝置封裝工具失敗。	無法建立套件，因為預期的作業系統元件遺失。

Android 錯誤

結束代碼	說明	備註
400	目前的 Android SDK 版本不支援特質。	檢查特質名稱的拼字是否正確，以及是否為所在元素的有效特質。如果特質在 Android 2.2 之後才出現，您可能需要在 ADT 命令中設定 -platformsdk 旗標。
401	目前的 Android SDK 版本不支援特質值。	檢查特質值的拼字是否正確，以及是否為特質的有效值。如果特質值在 Android 2.2 之後才出現，您可能需要在 ADT 命令中設定 -platformsdk 旗標。
402	目前的 Android SDK 版本不支援 XML 標記	檢查 XML 標記名稱的拼字是否正確，以及是否為有效的 Android 資訊清單文件元素。如果元素在 Android 2.2 之後才出現，您可能需要在 ADT 命令中設定 -platformsdk 旗標。
403	不允許覆寫 Android 標記	應用程式正嘗試覆寫保留給 AIR 使用的 Android 資訊清單元素。請參閱 第 65 頁「Android 設定」。
404	不允許覆寫 Android 特質	應用程式正嘗試覆寫保留給 AIR 使用的 Android 資訊清單特質。請參閱 第 65 頁「Android 設定」。
405	Android 標籤 %1 必須是 manifestAdditions 標籤中的第一個元素	將指定標籤移到必要的位置。
406	屬性 %1 (android 標籤 %2) 包含無效的值 %3。	請提供有效的屬性值。

ADT 環境變數

ADT 會讀取下列環境變數的值 (如果已設定)：

AIR_ANDROID_SDK_HOME 會指定 Android SDK 根目錄的路徑 (包含 tools 資料夾的目錄)。AIR 2.6+ SDK 包括實作相關 ADT 命令所需的 Android SDK 工具。只有當您想要使用不同版本的 Android SDK 時，才需要設定此值。如果已設定此變數，則在執行需要 -platformsdk 選項的 ADT 命令時，不需要指定 -platformsdk 選項。如果此變數和命令列選項均已設定，則會使用命令列上指定的路徑。

AIR_EXTENSION_PATH：指定用來搜尋應用程式所需之原生擴充功能的目錄清單。指定 ADT 命令列上的任何原生擴充功能目錄後，會依順序搜尋目錄清單。ADL 命令也會使用此環境變數。

備註：在某些電腦系統上，儲存在這些環境變數檔案系統路徑中的雙位元組字元可能會被錯譯。如果發生這個問題，請嘗試設定用來執行 ADT 的 JRE 以使用 UTF-8 字元集。而在 Mac 與 Linux 上啟動 ADT 的指令碼預設便已有此設定。在 Windows adt.bat 檔案中，或是當您從 Java 直接執行 ADT 時，請在 Java 命令列上指定 -Dfile.encoding=UTF-8 選項。

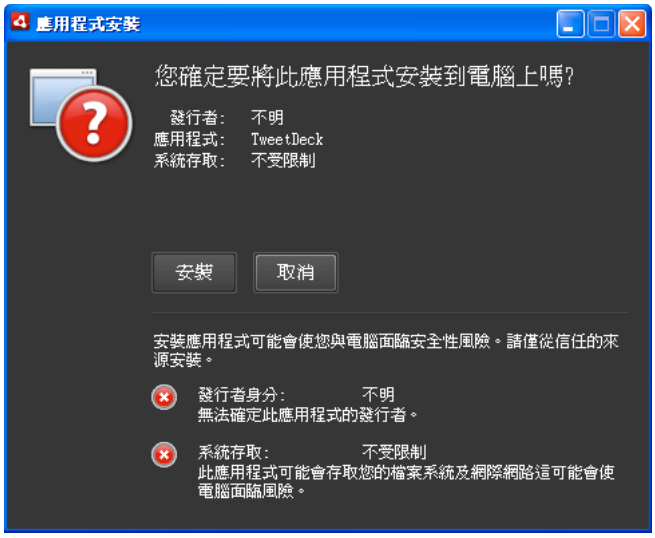
第 13 章 簽署 AIR 應用程式

為 AIR 檔加上數位簽名

使用公認的憑證授權單位 (CA) 所核發的憑證為 AIR 安裝檔案加上數位簽名後，可讓使用者確信他們所安裝的應用程式並未意外或惡意地遭到竄改，並能證實您本人具備簽署者 (發行者) 身分。如果 AIR 應用程式已經由受信任的憑證簽署，或已「鏈結」到安裝電腦上的受信任憑證，AIR 便會在安裝期間顯示發行者名稱：

已由受信任憑證簽署的應用程式安裝確認對話方塊

如果使用自我簽署憑證 (或未鏈結到受信任憑證的憑證) 來簽署應用程式，則使用者安裝您的應用程式時會承擔較高的安全性風險。安裝對話方塊會反映出此額外的風險：



已由自我簽署憑證簽署的應用程式安裝確認對話方塊

重要事項：意圖不良的組織或個人一旦取得您的簽署金鑰儲存檔案或察知您的私密金鑰，可能會假冒您的身分偽造 AIR 檔。

程式碼簽署憑證

程式碼簽署憑證之使用所涉及的安全保證、限制與法律義務等事項，刊載於核發商憑證授權單位所頒布的核證作業準則 (Certificate Practice Statements，CPS) 和用戶合約中。如需有關目前核發 AIR 程式碼簽署憑證的憑證授權單位合約的詳細資訊，請參閱：

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 用戶合約 (https://www.verisign.com/repository/subscriber/SUBAGR.html)

關於 AIR 程式碼簽署

AIR 檔一經簽署，便會為安裝檔案加上數位簽名。此簽名內含套件的摘要，用於證實 AIR 檔自簽署後未曾遭到竄改，且內含簽署憑證的相關資訊，用於驗證發行者身分。

AIR 使用已獲作業系統的憑證存放區支援的公開金鑰基礎結構 (PKI)，以確立憑證是否可受信任。安裝 AIR 應用程式的電腦必須直接信任用於簽署 AIR 應用程式的憑證，或間接信任將該憑證連結到受信任憑證授權單位的憑證鏈結，如此才能驗證發行者資訊。

如果 AIR 檔進行簽署所用的憑證未鏈結到其中一個受信任的根憑證 (這通常亦包括所有自我簽署的憑證)，即無法驗證發行者資訊。儘管 AIR 能夠判定 AIR 套件自簽署後未曾遭到竄改，但卻無從得知檔案的實際建立者和簽署者是誰。

備註：使用者可選擇信任自我簽署的憑證；以這類憑證進行簽署的 AIR 應用程式會將該憑證中 「一般名稱」欄位的值顯示成發行者名稱。AIR 並未提供任何方法可讓使用者將憑證認定為受信任。您必須另外提供憑證 (不包括私密金鑰) 給使用者，而使用者必須使用作業系統提供的機制或某種合適的工具，將憑證匯入系統憑證存放區中的適當位置。

關於 AIR 發行者識別名稱

重要事項：至於 AIR 1.5.3，不建議使用發行者 ID，而且不再以程式碼簽署憑證為基礎而進行計算。新的應用程式不需要，也不應該使用發行者 ID。更新現有的應用程式時，您必須在應用程式描述器檔案中指定原始發行者 ID。

AIR 1.5.3 之前的版本，在安裝 AIR 檔案的時候，AIR 應用程式安裝程式會產生發行者 ID。這是用於簽署 AIR 檔的憑證獨一無二的識別名稱。如果多個 AIR 應用程式重複使用同一份憑證，這些應用程式將收到相同的發行者 ID。使用其他憑證來簽署應用程式更新，或者有時候只是更新原始憑證的實體，都會變更發行者 ID。

在 AIR 1.5.3 以及更新的版本中，AIR 不會指派發行者 ID。透過 AIR 1.5.3 發行的應用程式，可以在應用程式描述器中指定發行者 ID。為原本發行為 AIR 1.5.3 以前版本的應用程式發行更新程式，才要指定發行 ID。如果您未在應用程式描述器中指定原始 ID，則新的 AIR 套件就不會被視為現有應用程式的更新程式。

若要判斷原始發行者 ID，請到原始應用程式安裝處的 META-INF/AIR 子目錄中找出 publisherid 檔案。這個檔案中的字串就是發行者 ID。應用程式描述器必須在應用程式描述器的命名空間宣告中指定 AIR 1.5.3 執行階段 (或更新版本)，以便手動指定發行者 ID。

發行者 ID (如果有的話) 可用於以下用途：

- 加密本機儲存區加密金鑰的一部分
- 應用程式儲存目錄路徑的一部分
- 本機連線之連線字串的一部分
- 識別字串的一部分，AIR 內部瀏覽器 API 用來呼叫應用程式
- OSID 的一部分 (建立自訂安裝 / 解除安裝程式時使用)

當發行者 ID 變更時，所有與此 ID 相關的 AIR 功能也會隨之變更行為。例如，現有加密本機儲存區中的資料將無法存取，而且任何建立該應用程式本機連線的 Flash 或 AIR 實體，都必須在連線字串中使用新的 ID。在 AIR 1.5.3 或更新版本中，無法變更已安裝之應用程式的發行者 ID。如果發行 AIR 套件時，您使用不同的發行者 ID，則安裝程式會將新的套件視為不同的應用程式而不是更新程式。

關於憑證格式

AIR 簽署工具接受任何可透過 Java 密碼架構 (JCA) 進行存取的金鑰儲存。這一類包括檔案類型的金鑰儲存如 PKCS12 格式檔案 (副檔名通常為 .pfx 或 .p12)、Java .keystore 檔案、PKCS11 硬體金鑰儲存，以及系統金鑰儲存。ADT 能夠存取的金鑰儲存格式取決於用來執行 ADT 的 Java Runtime 版本與組態而定。存取某些類型的金鑰儲存 (例如 PKCS11 硬體元件) 可能需要安裝及設定額外的軟體驅動程式和 JCA 外掛程式。

若要簽署 AIR 檔案，您可以使用大多數現有的程式碼簽署憑證，或者取得針對簽署 AIR 應用程式而明確核發的憑證。例如，由 VeriSign、Thawte、GlobalSign 或 ChosenSecurity 核發的下列任何類型憑證皆可使用：

- ChosenSecurity
  - Adobe AIR 的 TC 發行者 ID
- 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

備註：必須建立憑證以進行程式碼簽署。您不得使用 SSL 或其它類型的憑證簽署 AIR 檔案。

時間戳記

在簽署 AIR 檔時，封裝工具會向時間戳記授權單位的伺服器查詢，以取得可單獨驗證的簽署日期和時間。取得的時間戳記會內嵌於 AIR 檔中。只要簽署憑證在簽署之時仍屬有效，即使憑證已到期也能安裝 AIR 檔。另一方面，如果未能取得時間戳記，一旦憑證到期或遭撤銷即無法安裝 AIR 檔。

預設情況下，AIR 封裝工具會取得時間戳記。不過，若要在時間戳記服務無法使用時仍能封裝應用程式，您可以停用時間戳記。Adobe 建議所有公開散佈的 AIR 檔都應附上時間戳記。

AIR 封裝工具預設使用的時間戳記授權單位是 Geotrust。

取得憑證

為了取得憑證，您通常得造訪憑證授權單位的網站，並完成該公司的採購程序。應該使用哪些工具來產生 AIR 工具所需的金鑰儲存檔案，取決於購買的憑證類型，以及憑證儲存在接收端電腦上的方式而定，甚或某些情況下也和使用哪一種瀏覽器取得憑證有關。例如，若要從 Thawte 取得並匯出 Adobe Developer 憑證，您就必須使用 Mozilla Firefox。接著您就可以直接從 Firefox 使用者介面將憑證匯出成 .p12 或 .pfx 檔案。

備註：Java 1.5 版及以上版本在用來保護 PKCS12 憑證檔案的密碼中，不接受高位元 ASCII 字元。AIR 開發工具會使用 Java 來建立簽署的 AIR 套件。當您以 .p12 或 .pfx 檔案匯出憑證時，請僅使用一般 ASCII 字元做為密碼。

您可以使用專供封裝 AIR 安裝檔案用的 Air Development Tool (ADT) 產生自我簽署的憑證。部分的協力廠商工具亦可使用。

如需有關如何產生自我簽署憑證以及簽署 AIR 檔的指示，請參閱 第 142 頁「AIR Developer Tool (ADT)」。您也可以使用 Flash Builder、Dreamweaver 和 Flash 的 AIR 更新來匯出和簽署 AIR 檔案。

下列範例說明如何向 Thawte 憑證授權單位取得 AIR Developer Certificate 憑證，並備妥該憑證以搭配 ADT 使用。

範例：向 Thawte 取得 AIR Developer Certificate 憑證

備註：此範例所示者僅為取得與備妥適用的程式碼簽署憑證之諸多方法的其中一種。每個憑證授權單位都有其各自的政策和程序。

若要購得 AIR Developer Certificate 憑證，您必須使用 Mozilla Firefox 瀏覽器造訪 Thawte 網站。憑證的私密金鑰會儲存在瀏覽器的金鑰儲存內。請確認 Firefox 金鑰儲存已受到主控密碼的保護，而且電腦實體本身安全無虞 (一旦完成採購程序，您即可從瀏覽器的金鑰儲存內匯出憑證和私密金鑰，然後予以移除)。

在憑證註冊過程中，會產生私密 / 公開金鑰組。私密金鑰將自動儲存在 Firefox 金鑰儲存內。您必須使用同一部電腦和瀏覽器向 Thawte 網站申請及擷取憑證。

1. 造訪 Thawte 網站，然後瀏覽至 Code Signing Certificates 產品網頁。
2. 從 Code Signing Certificates 清單中選取 Adobe AIR Developer Certificate。
3. 完成註冊程序 (共 3 個步驟)。您必須提供公司資訊和聯絡資訊。Thawte 接著會執行身分驗證程序，而且可能要求您提供其它資訊。驗證完成後，Thawte 將寄給您一封電子郵件，並隨信附上有關如何擷取憑證的指示。

   備註： 您可在此找到各式必要書面文件的其它相關資訊：https://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/enroll_codesign_eng.pdf。
4. 向 Thawte 網站擷取核發的憑證。憑證將自動儲存至 Firefox 金鑰儲存。
5. 執行下列步驟，從 Firefox 金鑰儲存內匯出含有私密金鑰和憑證的金鑰儲存檔案：

   備註：從 Firefox 匯出的私密金鑰 / 憑證會匯出成 ADT、Flex、Flash 和 Dreamweaver 能夠使用的 .p12 (pfx) 格式。

   a. 開啟 Firefox「憑證管理員」對話方塊：
   b. 在 Windows 上：開啟「工具 > 選項 > 進階 > 加密 > 檢視憑證」
   c. 在 Mac OS 上：開啟「Firefox > 偏好設定 > 進階 > 加密 > 檢視憑證清單」
   d. 在 Linux 上：開啟「編輯 > 偏好設定 > 進階 > 加密 > 檢視憑證」
   e. 從憑證清單中選取 Adobe AIR Code Signing Certificate，再按一下「備份」按鈕。
   f. 輸入金鑰儲存檔案的檔案名稱以及匯出位置，然後按一下「儲存」。
   g. 如果您使用了 Firefox 主控密碼，瀏覽器會提示您輸入軟體安全裝置的密碼以便匯出檔案 (此密碼僅供 Firefox 使用)。
   h. 在「選擇憑證備份密碼」對話方塊中，建立金鑰儲存檔案的密碼。

      重要事項：此密碼可保護金鑰儲存檔案，且當您使用該檔案來簽署 AIR 應用程式時亦須提供此密碼。請務必選用安全的密碼。
   i. 按一下「確定」。您應該會收到密碼備份成功訊息。內含私密金鑰和憑證的金鑰儲存檔案將儲存成 .p12 副檔名 (為 PKCS12 格式)。
6. 將您所匯出的金鑰儲存檔案與 ADT、Flash Builder、Flash Professional 或 Dreamweaver 搭配使用。每次簽署 AIR 應用程式時，必須提供您為該檔案建立的密碼。

重要事項：私密金鑰和憑證仍將儲存在 Firefox 金鑰儲存內。這使您日後得以再匯出憑證檔的另一份副本，也提供了另一處基於維護憑證和私密金鑰的安全性而必須受到保護的存取點。

變更憑證

在某些情況下，您為 AIR 應用程式簽署更新程式所使用的憑證必須變更。這類情況包括：

- 更新原始簽署憑證
- 從自我簽署的憑證升級成憑證授權單位核發的憑證
- 將快要到期的自我簽署憑證變更成另一份憑證
- 從一個商業憑證變更為另一個商業憑證 (例如，當您的公司性質變更時)

要讓 AIR 將 AIR 檔案視為更新程式，您必須使用相同的憑證來簽署原始和更新的 AIR 檔案，或者將憑證移轉簽名套用至更新程式。移轉簽名是套用至使用原始憑證之 AIR 套件的第二個簽章。移轉簽名會使用原始憑證，來確立簽署者正是應用程式的原始發行者。

安裝內含移轉簽名的 AIR 檔案之後，新憑證會變成主要憑證。後續的更新程式就不需要移轉簽名。不過，只要能夠配合略過更新程式的使用者，您就應該套用移轉簽名。

重要事項：您必須變更憑證，並在原始憑證過期之前先使用原始憑證將移轉簽名套用至更新。否則，使用者就必須先解除安裝現有的應用程式版本，然後再安裝新版本。若為 AIR 1.5.3 或更新的版本，您可以在過期後的 365 天寬限期內，使用過期的憑證套用移轉簽名。不過，您無法使用過期的憑證來套用主要的應用程式簽名。

變更憑證：

1. 建立應用程式的更新版本
2. 使用 「新憑證」封裝並簽署更新版的 AIR 檔
3. 使用 「原始憑證」再簽署一次 AIR 檔 (利用 ADT -migrate 命令)

從其它方面來看，具有移轉簽名的 AIR 檔只是普通的 AIR 檔。若是在沒有原始版本的系統上安裝應用程式，AIR 將如往常般安裝新版本。

備註：在 AIR 1.5.3 之前，使用更新的憑證來簽署 AIR 應用程式時不一定需要移轉簽名。從 AIR 1.5.3 開始，更新的憑證一律需要移轉簽名。

若要套用移轉簽名，請使用 第 148 頁 「ADT migrate 命令」，如第 169 頁 「簽署 AIR 應用程式的更新版本」中所述。

備註：ADT migrate 命令無法與包含原生擴充功能的 AIR 桌上型應用程式一起搭配使用，因為那些應用程式是封裝為原生安裝程式，而非 .air 檔案。若要變更包含原生擴充功能之 AIR 桌上型應用程式的憑證，請搭配使用 第 143 頁 「ADT package 命令」與 -migrate 旗標來封裝應用程式。

應用程式識別變更

在 AIR 1.5.3 之前，安裝的更新程式若簽署移轉簽名，AIR 的識別會變更。變更應用程式的識別會有多種影響，其中包括：

- 新版應用程式無法存取現有加密本機儲存區中的資料。
- 應用程式儲存目錄的位置變更。舊有位置上的資料並未複製到新目錄 (但是新版應用程式仍可根據舊有的發行者 ID 找到原始目錄)。
- 應用程式無法再使用舊有的發行者 ID 來開啟本機連線。
- 網頁用來存取應用程式的識別字串會變更。
- 應用程式的 OSID 會變更 (編寫自訂安裝 / 解除安裝程式時會用到 OSID。)

發行 AIR 1.5.3 或更新版本的更新時，應用程式身分識別無法變更。您必須在更新 AIR 檔案的應用程式描述器中，指定原始應用程式和發行者 ID 。否則，新的套件不會視為更新程式。

備註：使用 AIR 1.5.3 (或更新版本) 發行新 AIR 應用程式時，請不要指定發行者 ID。

詞彙

本節提供一些您應該要瞭解的重要詞彙，以利於您決定如何簽署應用程式進行公開散佈。

詞彙	說明
憑證授權單位 (CA)	公開金鑰基礎結構網路中的機構，扮演著受信任協力廠商的角色，基本上負責驗證公開金鑰擁有者的身分。CA 通常會用自己的私密金鑰簽署並核發數位憑證，以證明確實驗證過憑證持有人的身分。
核證作業準則 (CPS)	憑證授權單位針對核發及驗證憑證所制定的作業細則與政策。CPS 是 CA 與其用戶及雙方信賴的第三者相互簽訂的合約之一部分。當中亦揭示了 CA 的身分驗證原則，及其核發之憑證所提供的保證等級。
憑證撤銷清單 (CRL)	核發後已遭撤銷而不該再受信任的憑證清單。AIR 將於 AIR 應用程式進行簽署時檢查 CRL；若是沒有時間戳記，則在應用程式安裝時還會檢查一次。
憑證鏈結	憑證鏈結即是一連串的憑證，此鏈結中的每一份憑證均已由下一份憑證進行簽署。
數位憑證	包含下列各項相關資訊的一份數位文件：擁有者的身分、擁有者的公開金鑰，以及憑證本身的身分識別。憑證授權單位核發的憑證是由隸屬於核發商 CA 本身的憑證進行簽署。
數位簽名	經過加密的訊息或摘要，唯有使用公開 / 私密金鑰組當中的公開金鑰才能予以解密。在 PKI 中，數位簽名包含了一份或多份數位憑證，這些憑證的來源最終可追溯到憑證授權單位。數位簽名可用來證實訊息 (或電腦檔案) 自簽署後未曾遭到竄改 (在密碼編譯演算法所提供的保證限度內)，且若是信任核發商憑證授權單位，亦可證實簽署者的身分。
金鑰儲存	包含數位憑證的資料庫，而在某些情況下還包含了相關的私密金鑰。
Java 密碼架構 (JCA)	一種可延伸的架構，用於管理及存取金鑰儲存。如需詳細資訊，請參閱「Java 密碼架構參考指南」。
PKCS #11	RSA 實驗室的密碼編譯元件介面標準 (Cryptographic Token Interface Standard)。此為硬體元件架構的金鑰儲存。
PKCS #12	RSA 實驗室的個人資訊交換語法標準 (Personal Information Exchange Syntax Standard)。此為檔案類型的金鑰儲存，通常內含私密金鑰及其相關的數位憑證。
私密金鑰	公開 / 私密金鑰組非對稱式密碼編譯系統當中的私密金鑰部分。私密金鑰絕不可洩露，更不該在網路上進行傳輸。加上數位簽名的訊息是由簽署者使用私密金鑰進行加密。
公開金鑰	公開 / 私密金鑰組非對稱式密碼編譯系統當中的公開金鑰部分。公開金鑰可公然發送，以便用於將私密金鑰所加密的訊息解密。
公開金鑰基礎結構 (PKI)	憑證授權單位據以證明公開金鑰擁有者身分的一套信任系統。網路用戶端仰賴受信任 CA 所核發的數位憑證來驗證數位訊息 (或檔案) 簽署者的身分。
時間戳記	加上數位簽名的數據，含有特定事件發生當時的日期和時間。ADT 能從 RFC 3161 相容時間伺服器取得時間戳記並加入到 AIR 套件中。如果有時間戳記，AIR 將使用此數據以確立憑證在簽署時的有效性。這樣一來，AIR 應用程式即使簽署憑證已到期仍能安裝。
時間戳記授權單位	核發時間戳記的授權單位。為了讓 AIR 能夠辨識，時間戳記必須符合 RFC 3161 標準，而且時間戳記簽名必須鏈結到安裝電腦上受信任的根憑證。

iOS 憑證

Apple 核發的程式碼簽署憑證可用來簽署 iOS 應用程式 (包括使用 Adobe AIR 開發的 iOS 應用程式)。若要在測試裝置上安裝應用程式，必須套用使用 Apple 開發憑證簽署的簽名。若要散佈已完成的應用程式，必須套用使用散發憑證簽署的簽名。

若要簽署應用程式，ADT 必須可以同時存取程式碼簽署憑證及關聯的私密金鑰。憑證檔案本身並不包括私密金鑰。您必須以同時包含憑證與私密金鑰的個人資訊交換檔案格式 (.p12 或 .pfx) 來建立金鑰儲存。請參閱第 167 頁「將開發人員憑證轉換成 P12 金鑰儲存檔案」。

產生憑證簽名要求

若要取得開發人員憑證，您必須先產生憑證簽名要求 (您將在 Apple iOS Provisioning Portal 送出此要求)。

憑證簽名要求程序會產生公開 / 私密金鑰組。私密金鑰會保留在您的電腦上。而您將傳送包含公開金鑰以及識別資訊的簽名要求給 Apple，Apple 在此扮演憑證授權單位的角色。Apple 會以自己的 World Wide Developer Relations 憑證簽署您的憑證。

在 **Mac OS** 產生憑證簽名要求

您可以在 Mac OS 利用 「鑰匙圈存取」應用程式產生代碼簽名要求。「鑰匙圈存取」應用程式位於 Applications 目錄下的 Utilities 子目錄。您可以在 Apple iOS Provisioning Portal 取得產生憑證簽名要求的指示。

在 **Windows** 產生憑證簽名要求

對 Windows 開發人員而言，最簡單的方法是取得 Mac 電腦上的 iPhone 開發人員憑證。不過，他們也可以在 Windows 電腦上取得憑證。首先，使用 OpenSSL 建立憑證簽名要求 (CSR 檔)：

1 在 Windows 電腦上安裝 OpenSSL (移至 http://www.openssl.org/related/binaries.html)。

   您可能也需要安裝 「Open SSL」下載頁面中所列出的 Visual C++ 2008 可轉散發套件檔案 (您不用在電腦安裝 Visual C++)。

2 開啟 Windows 命令工作階段，然後使用 CD 命令切換至 OpenSSL bin 目錄 (例如 c:\OpenSSL\bin\)。

3 在命令列輸入以下命令以建立專用密鑰：

```
openssl genrsa -out mykey.key 2048
```

   儲存此專用密鑰。您稍後將會用到它。

   使用 OpenSSL 時，請勿忽略錯誤訊息。OpenSSL 即使產生錯誤訊息，可能仍會輸出檔案。但這些檔案可能無法使用。如果發生錯誤，請檢查您的語法並重新執行命令。

4 在命令列輸入以下命令以建立 CSR 檔：

```
openssl req -new -key mykey.key -out CertificateSigningRequest.certSigningRequest  -subj
"/emailAddress=yourAddress@example.com, CN=John Doe, C=US"
```

   以您自己的值取代電子郵件地址、CN (憑證名稱) 及 C (國家 / 地區) 值。

5 將 CSR 檔上傳至 Apple 的 iPhone 開發人員網站 (請參閱 「申請 iPhone 開發人員憑證並建立佈建描述檔」)。

將開發人員憑證轉換成 P12 金鑰儲存檔案

若要建立 P12 金鑰儲存，您必須在單一檔案中組合 Apple 開發人員憑證與關聯的私密金鑰。建立金鑰儲存檔案的程序，需視您用以產生原始憑證簽名要求的方法以及儲存私密金鑰的位置而定。

在 **Mac OS** 將 **iPhone** 開發人員憑證轉換成 **P12** 檔案

從 Apple 下載 Apple iPhone 憑證之後，請將該憑證匯出為 P12 金鑰儲存格式。在 Mac® OS 執行此動作：

1 開啟 「鑰匙圈存取」應用程式 (位於 Applications/Utilities 資料夾)。

2 如果尚未將憑證新增至鑰匙圈，請選取 「檔案 > 輸入」。然後瀏覽至您向 Apple 取得的憑證檔案 (.cer 檔)。

3 在 「鑰匙圈存取」中選取 「鑰匙」類別。

4 選取與 「iPhone 開發憑證」相關的專用密鑰。

   專用密鑰由 iPhone 開發人員所指定：< 名字 > < 姓氏 > 與其配對的公用憑證。

5 按住 Command 鍵並按一下 iPhone 開發人員憑證，然後選取 「匯出 “iPhone 開發人員：名稱 ...”」。

6 以個人資訊交換 (.p12) 檔案格式儲存您的金鑰儲存。

7 系統將提示您建立密碼，以便在使用金鑰儲存簽署應用程式時，或是將此金鑰儲存中的金鑰與憑證傳輸到其他金鑰儲存時使用。

在 **Windows** 將 **Apple** 開發人員憑證轉換成 **P12** 憑證

若要開發 AIR for iOS 應用程式，您必須使用 P12 憑證檔案。您將根據從 Apple 收到的 Apple iPhone 開發人員憑證檔案來產生此憑證。

1 將您從 Apple 收到的開發人員憑證檔案轉換成 PEM 憑證檔案。從 OpenSSL bin 目錄執行以下命令列陳述式：

```
openssl x509 -in developer_identity.cer -inform DER -out developer_identity.pem -outform PEM
```

2 如果您使用的是 Mac 電腦上的鑰匙圈專用密鑰，請將它轉換成 PEM 密鑰：

```
openssl pkcs12 -nocerts -in mykey.p12 -out mykey.pem
```

3 現在您可以根據 iPhone 開發人員憑證的密鑰及 PEM 密鑰，產生有效的 P12 檔案：

```
openssl pkcs12 -export -inkey mykey.key -in developer_identity.pem -out iphone_dev.p12
```

如果您使用的是 Mac OS 鑰匙圈的密鑰，請使用上一個步驟所產生的 PEM 密鑰。否則，請使用先前產生的 OpenSSL 金鑰 (Windows)。

使用 ADT 建立未經簽署的 AIR 中繼檔案

您可以使用 -prepare 命令，建立未經簽署的 AIR 中繼檔案。AIR 中繼檔案必須在使用 ADT -sign 命令進行簽署之後，才能產生有效的 AIR 安裝檔案。

-prepare 命令會採用與 -package 命令所採用的相同旗標和參數 (簽署選項則例外)。唯一不同之處在於，輸出的是未經簽署的檔案。產生中繼檔案時，副檔名為 airi。

若要簽署 AIR 中繼檔案，請使用 ADT -sign 命令 (請參閱 第 148 頁 「ADT prepare 命令」)。

ADT -prepare 命令範例

```
adt -prepare unsignedMyApp.airi myApp.xml myApp.swf components.swc
```

使用 ADT 簽署 AIR 中繼檔案

若要使用 ADT 簽署 AIR 中繼檔案，請利用 -sign 命令。這個 sign 命令只適用於 AIR 中繼檔案 (副檔名為 airi)。一個 AIR 檔不能接受兩次簽署。

若要建立 AIR 中繼檔案，請使用 adt -prepare 命令 (請參閱 第 148 頁 「ADT prepare 命令」。)

簽署 **AIRI** 檔案

❖ 請使用 ADT -sign 命令和下列語法：

```
adt -sign SIGNING_OPTIONS airi_file air_file
```

SIGNING_OPTIONS：簽署選項會識別用於簽署 AIR 檔的私密金鑰和憑證。如需有關這些選項的詳細資訊，請參閱 第 153 頁 「ADT 程式碼簽署選項」。

airi_file：要簽署之 AIR 中繼檔案 (這個檔案尚未經過簽署) 的路徑。

air_file：要建立的 AIR 檔名稱。

ADT -sign 命令範例

```
adt -sign -storetype pkcs12 -keystore cert.p12 unsignedMyApp.airi myApp.air
```

如需詳細資訊，請參閱 第 148 頁 「ADT sign 命令」。

簽署 AIR 應用程式的更新版本

每次建立現有 AIR 應用程式的更新版本，都要簽署更新的應用程式。最佳的作法是，使用您用來簽署先前版本的相同憑證來簽署更新版本。如此，簽署就會與第一次的應用程式簽署完全相同。

如果用來簽署舊版應用程式的憑證已經過期，而且已更新或遭到取代，您可以使用已更新或全新 (取代項目) 的憑證來簽署更新版本。若要執行這項作業，請使用新憑證簽署應用程式，「並」使用原始憑證套用移轉簽名。移轉簽名會驗證更新是否由原始憑證擁有者所發佈。

在套用移轉簽名之前，請考慮下列要點：

- 若要套用移轉簽名，原始憑證必須仍然有效或者在過去 365 天內過期。這個期間稱為 「寬限期」，寬限期的時間長短在未來可能會有所變動。

  備註：在 AIR 2.6 之前，寬限期是 180 天。

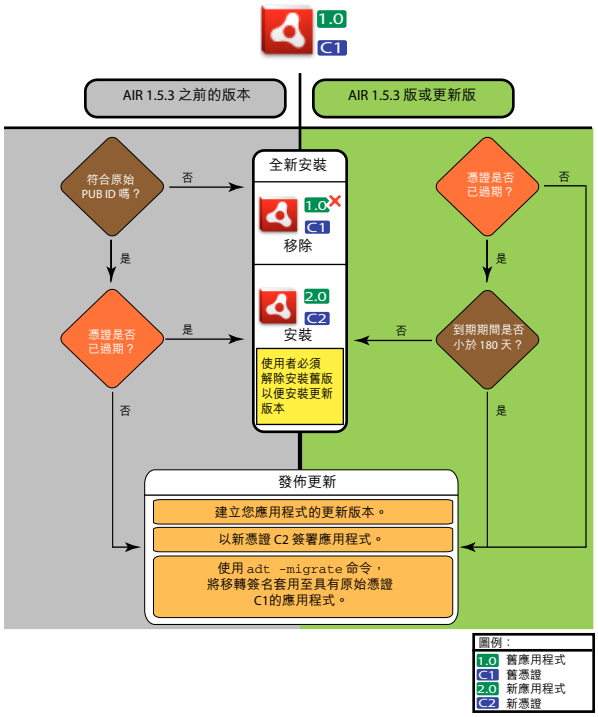
- 在憑證過期以及 365 天的寬限期之後，您便無法套用移轉簽名。在此情況下，使用者必須解除安裝現有版本，才能安裝更新版本。
- 365 天寬限期只適用於在應用程式描述器命名空間中，指定 AIR 1.5.3 版或更新版本的應用程式。

重要事項：以憑證過期的移轉簽名來簽署更新只是一種暫時性的解決方案。如需完整的解決方案，請建立標準化的簽署工作流程來管理應用程式更新的部署。例如，使用最新憑證來簽署每個更新，並且使用用來簽署舊版更新的憑證 (如果有的話) 來套用移轉憑證。將每個更新上傳到專屬的 URL，以便使用者下載應用程式。如需詳細資訊，請參閱第 223 頁 「應用程式更新的簽署工作流程」。

下列表格和圖表摘要說移轉簽名的工作流程：

案例	原始憑證的狀態	開發人員動作	使用者動作
以 Adobe AIR 執行階段 1.5.3 版或更新的版本為基礎的應用程式	有效	發佈 AIR 應用程式的最新版本。	不需要動作 應用程式會自動升級
	已過期，但仍在 365 天寬限期內	使用新憑證簽署應用程式。使用過期的憑證套用移轉簽名。	不需要動作 應用程式會自動升級
	已經過期且不在寬限期內	您無法將移轉簽名套用至 AIR 應用程式更新。 您必須使用新憑證來發佈另一個版本的 AIR 應用程式。解除安裝現有版本的 AIR 應用程式之後，使用者即可安裝新版本。	解除安裝目前版本的 AIR 應用程式，並安裝最新的版本

案例	原始憑證的狀態	開發人員動作	使用者動作
• 以 Adobe AIR 執行階段 1.5.2 版或較舊的版本為基礎的應用程式 • 更新的應用程式描述器中的發行者 ID 與先前版本的發行者 ID 相符	有效	發佈 AIR 應用程式的最新版本。	不需要動作 應用程式會自動升級
	已經過期且不在寬限期內	您無法將移轉簽名套用至 AIR 應用程式更新。 您必須使用新憑證來發佈另一個版本的 AIR 應用程式。解除安裝現有版本的 AIR 應用程式之後，使用者即可安裝新版本。	解除安裝目前版本的 AIR 應用程式，並安裝最新的版本
• 以 Adobe AIR 執行階段 1.5.2 版或較舊的版本為基礎的應用程式 • 更新的應用程式描述器中的發行者 ID 與先前版本的發行者 ID 「不符」	任意	使用有效的憑證簽署 AIR 應用程式並發佈 AIR 應用程式的最新版本	解除安裝目前版本的 AIR 應用程式，並安裝最新的版本



更新的簽署工作流程

將 AIR 應用程式移轉為使用新憑證

若要在更新應用程式的同時將 AIR 應用程式移轉為新憑證：

1 建立應用程式的更新版本

2 使用 「新憑證」封裝並簽署更新版的 AIR 檔

3 使用 -migrate 命令，以 「原始」憑證再次簽署 AIR 檔

使用 -migrate 命令簽署的 AIR 檔案除了用來更新任何以舊版憑證簽署的先前版本之外，也可以用來安裝新版的應用程式。

備註：更新使用 1.5.3 以前的 AIR 版本發行的應用程式時，您必須在應用程式描述器中指定原始發行者 ID。否則，應用程式的使用者必須解除安裝舊的版本，才能安裝更新程式。

請使用 ADT -migrate 命令和下列語法：

```
adt -migrate SIGNING_OPTIONS air_file_in air_file_out
```

- **SIGNING_OPTIONS**：簽署選項會識別用於簽署 AIR 檔的私密金鑰和憑證。這些選項必須識別 「原始」簽署憑證。如需有關這些選項的詳細資訊，請參閱 第 153 頁 「ADT 程式碼簽署選項」。
- **air_file_in**：本身為更新程式的 AIR 檔，已經使用 「新的」憑證加以簽署。
- **air_file_out**：要建立的 AIR 檔。

備註：用於輸入和輸出 AIR 檔案的檔案名稱必須不同。

下列範例示範使用 -migrate 旗標呼叫 ADT，以便將移轉簽名套用至 AIR 應用程式的更新版本：

```
adt -migrate -storetype pkcs12 -keystore cert.p12 myAppIn.air myApp.air
```

備註：在 AIR 1.1 發行版本中，已經將 -migrate 命令加入至 ADT。

將原生安裝程式 AIR 應用程式移轉為使用新憑證

以原生安裝程式形式發佈的 AIR 應用程式 (例如，使用原生擴充功能 API 的應用程式) 無法透過 ADT -migrate 命令進行簽署，因為它是平台特定的原生應用程式，而非 .air 檔案。相反地，若要將以原生擴充功能形式發佈的 AIR 應用程式移轉為新憑證：

1. 建立應用程式的更新版本。
2. 確認在您的應用程式描述器 (app.xml) 檔案中，<supportedProfiles> 標籤同時包含桌上型描述檔和 extendedDesktop 描述檔 (或者從應用程式描述器中移除 <supportedProfiles> 標籤)。
3. 使用 ADT -package 命令搭配 「新」憑證，將更新應用程式封裝為 「.air 檔案」並加以簽署。
4. 使用 ADT -migrate 命令搭配 「原始」憑證，將移轉憑證套用至 .air 檔案 (如第 170 頁 「將 AIR 應用程式移轉為使用新憑證」中所述)。
5. 使用 ADT -package 命令搭配 -target native 旗標，將 .air 檔案封裝成原生安裝程式。由於應用程式已經簽署，因此，在這個步驟中，您不需要指定簽署憑證。

下列範例示範這個程序的步驟 3-5。此程式碼會以 -package 命令呼叫 ADT、以 -migrate 命令呼叫 ADT，然後再次以 -package 命令呼叫 ADT，以便將 AIR 應用程式的更新版本封裝為原生安裝程式：

```
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
```

將使用原生擴充功能的 AIR 應用程式移轉為使用新憑證

使用原生擴充功能的 AIR 應用程式無法透過 ADT -migrate 命令進行簽署，也不能透過用來移轉原生安裝程式 AIR 應用程式的程序進行移轉，因為它無法發佈為中繼 .air 檔案。相反地，若要將使用原生擴充功能的 AIR 應用程式移轉為新憑證：

1. 建立應用程式的更新版本
2. 使用 ADT -package 命令來封裝並簽署更新原生安裝程式。以 「新」憑證封裝應用程式，並且包含指定 「原始」憑證的 -migrate 旗標。

使用下列語法呼叫含有 -migrate 旗標的 ADT -package 命令：

```
adt -package AIR_SIGNING_OPTIONS -migrate MIGRATION_SIGNING_OPTIONS -target package_type
NATIVE_SIGNING_OPTIONS output app_descriptor FILE_OPTIONS
```

- **AIR_SIGNING_OPTIONS**：簽署選項會識別用於簽署 AIR 案的私密金鑰和憑證。這些選項會識別「新」簽署憑證。如需有關這些選項的詳細資訊，請參閱 第 153 頁「ADT 程式碼簽署選項」。
- **MIGRATION_SIGNING_OPTIONS**：簽署選項會識別用於簽署 AIR 檔案的私密金鑰和憑證。這些選項會識別「原始」簽署憑證。如需有關這些選項的詳細資訊，請參閱 第 153 頁「ADT 程式碼簽署選項」。
- 其他選項與用於封裝原生安裝程式 AIR 應用程式的選項相同。如需詳細資訊，請參閱 第 143 頁「ADT package 命令」。

下列範例示範如何使用 -package 命令和 -migrate 旗標呼叫 ADT，以便封裝使用原生擴充功能的 AIR 應用程式更新版本，並且將移轉簽名套用至更新：

```
adt -package -storetype pkcs12 -keystore new_cert.p12 -migrate -storetype pkcs12 -keystore original_cert.p12
-target native myApp.exe myApp.xml myApp.swf
```

備註：-package 命令的 -migrate 旗標可以在 AIR 3.6 及更新版本的 ADT 中取得。

使用 ADT 建立自我簽署的憑證

您可以使用自我簽署的憑證來產生有效的 AIR 安裝檔案。但是，自我簽署的憑證只能為您的使用者提供有限的安全保證。因此無法確保自我簽署憑證的可靠性。安裝具有自我簽署憑證的 AIR 檔時，使用者所看到的發行者資訊會顯示為「不明」。ADT 產生的憑證有效期限為五年。

如果您為使用自我產生之憑證簽署的 AIR 應用程式建立更新程式，就必須使用相同的憑證來簽署原始和更新的 AIR 檔。ADT 所產生的憑證都具有唯一性，即使是使用相同的參數也一樣。因此，如果您要使用 ADT 產生的憑證來自我簽署更新程式，請將原始憑證保存在安全的位置。此外，當 ADT 產生的原始憑證過期之後，您就無法再產生更新的 AIR 檔 (您可以使用不同的憑證發行新應用程式，但是不能針對相同的應用程式發行新版本)。

重要事項：基於自我簽署憑證的限制，Adobe 強烈建議針對公開發行的 AIR 應用程式，使用具聲譽的憑證授權單位發行的商業憑證。

ADT 產生的憑證和相關私密金鑰會儲存在類型為 PKCS12 的金鑰儲存檔中。指定的密碼是在金鑰上設定，而非金鑰儲存。

憑證產生範例

```
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
```

若要使用這些憑證來簽署 AIR 檔，請使用下列簽署選項並搭配 ADT -package 或 -prepare 命令：

```
-storetype pkcs12 -keystore newcert.p12 -storepass 39#wnetx3tl
-storetype pkcs12 -keystore SigningCert.p12 -storepass 39#wnetx3tl
```

備註：Java 1.5 版及以上版本在用來保護 PKCS12 憑證檔案的密碼中，不接受高位元 ASCII 字元。請僅使用一般 ASCII 字元做為密碼。

第 14 章 AIR 應用程式描述器檔案

每個 AIR 應用程式都需要應用程式描述器檔案。應用程式描述器檔案是一種 XML 文件，可以用來定義應用程式的基本屬性。

在您建立專案時，許多支援 AIR 的開發環境會自動產生應用程式描述器。若無產生，您必須建立自行建立描述器檔案。AIR 和 Flex SDK 的 samples 目錄中提供一個樣本描述器檔案 descriptor-sample.xml。

您可以使用任何檔名做為應用程式描述器檔案的名稱。當您封裝應用程式時，應用程式描述器檔案會更名為 application.xml，並放置在 AIR 套件內部的特定目錄中。

範例應用程式描述器

下列的應用程式描述器文件會設定大部分 AIR 應用程式所使用的基本屬性：

```
<?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>
```

如果應用程式使用 HTML 檔案 (而不是 SWF 檔) 當做本身的根內容時，則只有 <content> 元素會有所不同。

```
<content>
    HelloWorld.html
</content>
```

應用程式描述器變更

下列 AIR 版本中的 AIR 應用程式描述器已變更。

AIR 1.1 描述器變更

允許使用 text 元素來當地語系化應用程式的 name 與 description 元素。

AIR 1.5 描述器變更

contentType 已變成 fileType 的必要子系。

AIR 1.5.3 描述器變更

新增 publisherID 元素，允許應用程式指定發行者 ID 值。

AIR 2.0 描述器變更

新增：

- aspectRatio
- autoOrients
- fullScreen
- image29x29 (請參閱 imageNxN)
- image57x57
- image72x72
- image512x512
- iPhone
- renderMode
- supportedProfiles

AIR 2.5 描述器變更

移除：version

新增：

- android
- extensionID
- extensions
- image36x36 (請參閱 imageNxN)
- manifestAdditions
- versionLabel
- versionNumber

AIR 2.6 描述器變更

新增：

- image114x114 (請參閱 imageNxN)
- requestedDisplayResolution
- softKeyboardBehavior

AIR 3.0 描述器變更

新增：

- colorDepth
- direct 做為 renderMode 的有效值
- 在桌上型平台中不再忽略 renderMode
- 可以指定 Android <uses-sdk> 元素 (先前是不允許的)。

AIR 3.1 描述器變更

新增：

- 第 186 頁「Entitlements」

AIR 3.2 描述器變更

新增：

- depthAndStencil
- supportedLanguages

AIR 3.3 描述器變更

新增：

- aspectRatio 現在包括 ANY 選項。

AIR 3.4 描述器變更

新增：

- image50x50 (請參閱 第 193 頁「imageNxN」)
- image58x58 (請參閱 第 193 頁「imageNxN」)
- image100x100 (請參閱 第 193 頁「imageNxN」)
- image1024x1024 (請參閱 第 193 頁「imageNxN」)

AIR 3.6 描述器變更

新增：

- 第 197 頁「iPhone」元素中的 第 203 頁「requestedDisplayResolution」現在包括 excludeDevices 特質，可讓您指定使用高或標準解析度的 iOS 目標
- 第 195 頁「initialWindow」中的 第 203 頁「requestedDisplayResolution」新元素會指定是否在桌上型平台 (例如具有高解析度顯示器的 Mac) 上使用高或標準解析度

AIR 3.7 描述器變更

新增：

- 第 197 頁「iPhone」元素現在提供 第 187 頁「externalSwfs」元素，可讓您指定要在執行階段載入的 SWF 清單。

- 第 197 頁「iPhone」元素現在提供 第 191 頁「forceCPURenderModeForDevices」元素，可讓您針對指定的裝置組合強制套用 CPU 顯示模式。

應用程式描述器檔案結構

應用程式描述器檔案是一種 XML 文件，具有下列結構：

```
<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>
    < 第 204 頁「supportedLanguages」 >...</ 第 204 頁「supportedLanguages」 >
    <supportedProfiles>...</supportedProfiles>
    <versionNumber>...</versionNumber>
    <versionLabel>...</versionLabel>
</application>
```

AIR 應用程式描述器元素

下列元素的字典會說明 AIR 應用程式描述器檔案的每個法律元素。

allowBrowserInvocation

Adobe AIR 1.0 及更新的版本 — 選擇性

啟用 AIR 內部瀏覽器 API 來偵測和啟動應用程式。

如果將這個值設定為 true，請務必考量安全隱憂。這些隱憂將描述於從瀏覽器叫用 AIR 應用程式 (適用於 ActionScript 開發人員) 和 Invoking an AIR application from the browser (適用於 HTML 開發人員)。

如需詳細資訊，請參閱第 220 頁「從瀏覽器啟動已安裝的 AIR 應用程式」。

父元素：第 178 頁「application」

子元素：無

內容
true 或 false (預設值)

範例

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

android

Adobe AIR 2.5 及更新的版本 — 選擇性

允許您將元素新增至 Android 資訊清單檔案。AIR 會為每個 APK 套件建立 AndroidManifest.xml 檔案。您可以使用 AIR 應用程式描述器中的 android 元素來新增其他項目。Android 以外的平台會忽略此元素。

父元素：第 178 頁「application」

子元素：

- 第 182 頁「colorDepth」
- 第 198 頁「manifestAdditions」

內容
定義 Android 特定屬性的元素，這些元素將新增至 Android 應用程式資訊清單。

範例

```
<android>
    <manifestAdditions>
        ...
    </manifestAdditions>
</android>
```

更多說明主題

第 65 頁「Android 設定」

AndroidManifest.xml 檔案

application

Adobe AIR 1.0 及更新的版本 — 必要

AIR 應用程式描述器文件的根元素。

父元素：無

子元素：

- 第 177 頁「allowBrowserInvocation」
- 第 177 頁「android」
- 第 183 頁「copyright」
- 第 184 頁「customUpdateUI」
- 第 185 頁「description」
- 第 187 頁「extensions」
- 第 188 頁「filename」
- 第 189 頁「fileTypes」
- 第 192 頁「icon」
- 第 193 頁「id」
- 第 195 頁「initialWindow」
- 第 196 頁「installFolder」
- 第 197 頁「iPhone」
- 第 200 頁「name」
- 第 201 頁「programMenuFolder」
- 第 202 頁「publisherID」

- 第 204 頁「supportedLanguages」
- 第 205 頁「supportedProfiles」
- 第 207 頁「version」
- 第 208 頁「versionLabel」
- 第 208 頁「versionNumber」

特質

minimumPatchLevel — 此應用程式所需的 AIR 執行階段最低修補程式層級。

xmlns — XML 命令空間特質會決定應用程式所需的 AIR 執行階段版本。

這個命名空間會隨著 AIR 的每一個主要發行版本變更 (但是發行次要修補程式時則不會)。命名空間的最後一個區段 (例如「3.0」) 表示應用程式所需的執行階段版本。

主要 AIR 版本的 xmlns 值為：

```
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"
```

對於 SWF 類型的應用程式，指定在應用程式描述器中的 AIR 執行階段版本會決定可載入為應用程式初始內容的 SWF 最高版本。指定 AIR 1.0 或 AIR 1.1 的應用程式只能使用 SWF9 (Flash Player 9) 檔當作初始內容，即使使用 AIR 2 執行階段執行，也是如此。指定 AIR 1.5 (或更新版本) 的應用程式可以使用 SWF9 或 SWF10 (Flash Player 10) 檔案當作初始內容。

SWF 版本會決定可使用的 AIR 和 Flash Player API 版本。如果 SWF9 檔用來當做 AIR 1.5 應用程式的初始內容，則該應用程式只能存取 AIR 1.1 和 Flash Player 9 API。另外，在現有 AIR 2.0 或 Flash Player 10.1 中 的 API 行為變更將不會生效 (對 API 所做的與安全性相關的重要變更則不受此原則的限制，而且可追溯套用到執行階段目前或未來的修補程式)。

對於 HTML 類型的應用程式，應用程式描述器中指定的執行階段版本會決定應用程式可用的 AIR 和 Flash Player API 的版本。HTML、CSS 和 JavaScript 的行為永遠會由所安裝 AIR 執行階段中使用的 Webkit 版本來決定，而不會由應用程式描述器決定。

當 AIR 應用程式載入 SWF 內容時，可供內容使用的 AIR 和 Flash Player API 版本會根據內容的載入方法而定。有效版本有時候是由應用程式描述器命名空間決定，有時候是由正在載入之內容的版本決定，有時候則是由已經載入之內容的版本決定。下表顯示如何根據載入方法決定 API 版本：

載入內容的方法	決定 API 版本的方法
初始內容，SWF 類型應用程式	「已載入」檔案的 SWF 版本
初始內容，HTML 類型應用程式	應用程式描述器命名空間
由 SWF 內容載入的 SWF	「正在載入」之內容的版本

載入內容的方法	決定 API 版本的方法
由 HTML 內容使用 <script> 標籤所載入的 SWF 元件庫	應用程式描述器命名空間
由 HTML 內容使用 AIR 或 Flash Player API (例如 flash.display.Loader) 所載入的 SWF	應用程式描述器命名空間
由 HTML 內容使用 <object> 或 <embed> 標籤 (或對應的 JavaScript API) 所載入的 SWF	「已載入」檔案的 SWF 版本

當載入的 SWF 檔案版本與載入內容的版本不同時，可能會遇到兩個問題：

- 透過舊版 SWF 載入新版 SWF — 載入內容中將無法解析新版 AIR 和 Flash Player 所新增 API 的參照
- 透過新版 SWF 載入舊版 SWF — 新版 AIR 和 Flash Player 中變更的 API 可能會出現載入內容無法預期的行為。

內容

應用程式元素包含定義 AIR 應用程式屬性的子元素。

範例

```
<?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 及更新的版本、**iOS** 與 **Android** — 選擇性

指定應用程式的外觀比例。

如果未指定，則應用程式會以裝置的「自然」外觀比例和方向開啟。自然方向因裝置而異。在小螢幕裝置 (例如手機) 上通常是縱向比例。在某些裝置上，例如 iPad 平板電腦，應用程式會以目前方向開啟。在 AIR 3.3 及更新版本中，這會套用至整個應用程式，而不只是一開始的顯示。

父元素：第 195 頁「initialWindow」

子元素：無

內容

portrait、landscape 或 any

範例

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

autoOrients

Adobe AIR 2.0 及更新的版本、**iOS** 與 **Android** — 選擇性

指定應用程式的內容方向是否會隨著裝置的實際方向自動變換方向。如需詳細資訊，請參閱舞台方向。

使用自動調整方向時，請考慮將舞台的 align 和 scaleMode 屬性設為下列值：

```
stage.align = StageAlign.TOP_LEFT;
stage.scaleMode = StageScaleMode.NO_SCALE;
```

這些設定可讓應用程式以左上角為中心旋轉，並避免應用程式內容自動縮放。其他縮放模式雖然會調整內容來符合旋轉的舞台尺寸，但同時也會裁切、扭曲或過度縮小該內容。由您自行重繪或重放內容，通常比較可以得到好結果。

父元素：第 195 頁「initialWindow」

子元素：無

內容
true 或 false (預設值)

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

colorDepth

Adobe AIR 3 及更新的版本 — 選擇性

指定使用 16 位元或 32 位元色彩。

使用 16 位元色彩可提高顯示效能，但會　牲顏色精確度。在 AIR 3 之前，Android 上一向使用 16 位元色彩。在 AIR 3 中，預設使用 32 位元色彩。

備註：如果您的應用程式使用 StageVideo 類別，則必須使用 32 位元色彩。

父元素：第 177 頁「android」

子元素：無

內容
下列其中一個值：

- 16 位元
- 32 位元

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

containsVideo

指定應用程式是否會包含任何視訊內容。

父元素：第 177 頁「android」

子元素：無

內容
下列其中一個值：

- true
- False

範例
```
<android>
    <containsVideo>true</containsVideo>
    <manifestAdditions>...</manifestAdditions>
</android>
```

content

Adobe AIR 1.0 及更新的版本 — 必要

為 content 元素指定的值是應用程式之主要內容檔案的 URL。這個檔案可能會是 SWF 檔或 HTML 檔。指定的 URL 必須相對於應用程式安裝資料夾的根目錄 (使用 ADL 執行 AIR 應用程式時，指定的 URL 則必須相對於包含應用程式描述器檔案的資料夾。您可以使用 ADL 的 root-dir 參數，指定不同的根目錄)。

父元素：第 195 頁 「initialWindow」

子元素：無

內容
相對於應用程式目錄的 URL。由於 content 元素的值會視為 URL，所以根據 RFC 1738 中定義的規則，必須將內容檔案名稱中的字元以 URL 編碼方式加以處理。例如，空格字元就必須編碼為 %20。

範例

```
<content>TravelPlanner.swf</content>
```

contentType

Adobe AIR 1.0 到 **1.1** — 選擇性；**AIR 1.5** 及更新的版本 — 必要

contentType 在 AIR 1.5 中為必要項目 (在 AIR 1.0 和 1.1 中則為選擇性項目)。此屬性會協助某些作業系統找到最適合開啟檔案的應用程式。其值必須是檔案內容的 MIME 類型。請注意，如果檔案類型已登錄且已指定為 MIME 類型，則在 Linux 上會忽略此 。

父元素：第 189 頁 「fileType」

子元素：無

內容
MIME 類型與子類型。如需有關 MIME 類型的詳細資訊，請參閱 RFC2045。

範例

```
<contentType>text/plain</contentType>
```

copyright

Adobe AIR 1.0 及更新的版本 — 選擇性

AIR 應用程式的版權資訊。在 Mac OS 中，所安裝應用程式的版權內容會出現在 「關於」對話方塊中。此外，該應用程式之 Info.plist 檔案的 NSHumanReadableCopyright 欄位中也會使用版權資訊。

父元素：第 178 頁 「application」

子元素：無

內容
包含應用程式版權資訊的字串。

範例

```
<copyright>© 2010, Examples, Inc.All rights reserved.</copyright>
```

customUpdateUI

Adobe AIR 1.0 及更新的版本 — 選擇性

指出應用程式是否將提供自己的更新對話方塊。如果為 false，則 AIR 會向使用者顯示標準更新對話方塊。只有以 AIR 檔案散佈的應用程式才能使用內建 AIR 更新系統。

當已安裝應用程式版本的 customUpdateUI 元素設定為 true，而且使用者稍後按兩下代表新版本的 AIR 檔或透過隱藏安裝功能安裝應用程式的更新程式時，執行階段就會開啟已安裝的應用程式版本。執行階段不會開啟預設的 AIR 應用程式安裝程式。接著，應用程式邏輯便會決定更新作業的執行方式 (AIR 檔中的應用程式 ID 和發行者 ID 都必須符合已安裝應用程式中的值，才能執行升級作業)。

備註：只有在已安裝應用程式，以及使用者按兩下包含更新程式的 AIR 安裝檔案，或透過隱藏安裝功能安裝應用程式之更新程式的情況下，customUpdateUI 機制才會發揮效用。不管 customUpdateUI 是否為 true，您都可以透過專屬應用程式邏輯下載並啟動更新程式，並且視需要顯示自訂的 UI。

如需詳細資訊，請參閱第 222 頁 「更新 AIR 應用程式」。

父元素：第 178 頁 「application」

子元素：無

內容
true 或 false (預設值)

範例

```
<customUpdateUI>true</customUpdateUI>
```

depthAndStencil

Adobe AIR 3.2 及更新的版本 — 選擇性

指出應用程式需要使用深度或模板緩衝區。您通常會在使用 3D 內容時使用這些緩衝區。根據預設，這個元素的值為 false，會停用深度和模板緩衝區。這是必要元素，因為緩衝區必須在任何內容載入之前，於應用程式啟動時配置完成。

這個元素的設定必須符合傳遞做為 Context3D.configureBackBuffer() 方法之 enableDepthAndStencil 引數的值。如果值不符，AIR 就會發出錯誤。

這個元素只能在 renderMode = direct 時使用。如果 renderMode 不等於 direct，ADT 就會擲回錯誤 118：

```
<depthAndStencil> element unexpected for render mode cpu.  It requires "direct" render mode.
```

父元素：第 195 頁 「initialWindow」

子元素：無

內容
true 或 false (預設值)

範例

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

description

Adobe AIR 1.0 及更新的版本 — 選擇性

應用程式的說明，會顯示在 AIR 應用程式安裝程式中。

如果您指定單一文字節點 (而非多個 text 元素)，則不論系統語言為何，AIR 應用程式安裝程式都會使用這項說明。否則，AIR 應用程式安裝程式會使用最符合使用者作業系統之使用者介面語言的說明。例如，假設正在執行一項安裝作業，其中應用程式描述器檔案的 description 元素包含代表 en (英文) 地區的值。如果使用者的系統將 en (英文) 識別為使用者介面語言，則 AIR 應用程式安裝程式會使用 en 說明。此外，如果系統使用者介面語言為 en-US (美式英文)，安裝程式也會使用 en 說明。但是，如果系統使用者介面語言為 en-US，而且應用程式描述器檔案定義了 en-US 和 en-GB 這兩個名稱，那麼 AIR 應用程式安裝程式就會使用 en-US 這個值。如果應用程式定義的說明都不符合系統使用者介面語言，AIR 應用程式安裝程式就會使用在應用程式描述器檔案中定義的第一個 description 值。

如需開發多國語言應用程式的相關資訊，請參閱第 254 頁 「當地語系化 AIR 應用程式」。

父元素：第 178 頁 「application」

子元素：第 206 頁 「text」

內容
在 AIR 1.0 應用程式描述器結構中，只允許為該名稱 (而非多個 text 元素) 定義一個簡單文字節點。

在 AIR 1.1 (或更新版本) 中，您可以在 description 元素內指定多個語言。如 RFC4646 (http://www.ietf.org/rfc/rfc4646.txt) 中所定義，每一個 text 元素的 xml:lang 特質都可以指定一個語言代碼。

範例
具有簡單文字節點的說明：

```
<description>This is a sample AIR application.</description>
```

以英文、法文和西班牙文當地語系化的文字元素說明 (適用於 AIR 1.1 及更新的版本)：

```
<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 及更新的版本 — 必要

由作業系統向使用者顯示的檔案類型說明。檔案類型說明無法當地語系化。

請參閱：做為應用程式元素子系的 第 185 頁 「description」

父元素：第 189 頁 「fileType」

子元素：無

內容
說明檔案內容的字串。

範例

```
<description>PNG image</description>
```

embedFonts

可讓您在 AIR 應用程式的 StageText 上使用自訂字體。這個元素是選擇性的。

父元素：第 178 頁「application」

子元素：第 190 頁「字體」

內容
embedFonts 元素可包含任何數目的字體元素。

範例

```
<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 及更新的版本 — 僅 **iOS**，選擇性

iOS 使用名為 Entitlements 的屬性，提供應用程式存取額外的資源和功能。使用 Entitlements 元素即可在行動 iOS 應用程式中指定這項資訊。

父元素：第 197 頁「iPhone」

子元素：iOS Entitlements.plist 元素

內容
包含子元素，這些子元素會指定索引鍵值配對，做為應用程式的 Entitlements.plist 設定。Entitlements 元素的內容應該包括在 CDATA 區塊中。如需詳細資訊，請參閱 iOS Developer Library 中的 權益索引鍵參考。

範例

```
<iphone>
...
   <Entitlements>
      <![CDATA[
         <key>aps-environment</key>
         <string>development</string>
      ]]>
   </Entitlements>
</iphone>
```

extension

Adobe AIR 1.0 及更新的版本 — 必要

檔案類型的副檔名字串。

父元素：第 189 頁「fileType」

子元素：無

內容
識別副檔名字元的字串 (不包含點 “.”)。

範例
```
<extension>png</extension>
```

extensionID

Adobe AIR 2.5 及更新的版本

指定應用程式使用之 ActionScript 擴充功能的 ID。ID 會定義在副檔名描述器文件中。

父元素：第 187 頁 「extensions」

子元素：無

內容
ActionScript 擴充功能 ID 的識別字串。

範例
```
<extensionID>com.example.extendedFeature</extensionID>
```

extensions

Adobe AIR 2.5 及更新的版本 — 選擇性

識別應用程式使用的 ActionScript 擴充功能。

父元素：第 178 頁 「application」

子元素：第 187 頁 「extensionID」

內容
extensionID 子元素，包含擴充功能描述器檔案中的 ActionScript 擴充功能 ID。

範例
```
<extensions>
    <extensionID>extension.first</extensionID>
    <extensionID>extension.next</extensionID>
    <extensionID>extension.last</extensionID>
</extensions>
```

externalSwfs

Adobe AIR 3.7 及更新的版本、僅 **iOS** — 選擇性

指定文字檔的名稱，檔案內含要由 ADT 設定用於遠端裝載的 SWF 清單。您可以封裝應用程式所使用的 SWF 子集，並且使用 Loader.load() 方法在執行階段載入其餘 (僅限資源) 的外部 SWF，藉此將初始應用程式下載大小減到最小。若要使用這個功能，您必須封裝應用程式，如此，ADT 就會將所有 ActionScript ByteCode (ABC) 從外部載入的 SWF 檔案移至主應用程式 SWF，留下只包含資源的 SWF 檔案。這是為了符合 Apple Store 禁止在安裝應用程式之後下載任何程式碼的規則所致。

如需詳細資訊，請參閱第 74 頁 「載入僅限資源的外部 SWF，將下載大小減到最小」。

父元素：第 197 頁 「iPhone」、第 195 頁 「initialWindow」

子元素：無

內容
文字檔的名稱，檔案內含要遠端裝載、以行分隔的 SWF 清單。

特質：
無。

範例
iOS：

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

filename

Adobe AIR 1.0 及更新的版本 — 必要

安裝應用程式時，要當做應用程式的檔名 (不含副檔名) 使用的字串。這個應用程式檔案會在執行階段啟動 AIR 應用程式。如果沒有指定 name 的值，那麼也會使用 filename 當做安裝資料夾的名稱。

父元素：第 178 頁 「application」

子元素：無

內容
filename 屬性可以包含下列字元以外的任何 Unicode (UTF-8) 字元，各種檔案系統都會禁止將下列字元當做檔名使用：

字元	十六進位碼
右列十六進位碼範圍內的字元	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

filename 的值不能以半形的點 (.) 做為結尾。

範例

```
<filename>MyApplication</filename>
```

fileType

Adobe AIR 1.0 及更新的版本 — 選擇性

說明應用程式可以註冊的單一檔案類型。

父元素：第 189 頁「fileTypes」

子元素：

- 第 183 頁「contentType」
- 第 185 頁「description」
- 第 186 頁「extension」
- 第 192 頁「icon」
- 第 201 頁「name」

內容
說明檔案類型的元素。

範例

```
<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 及更新的版本 — 選擇性

您可以使用 fileTypes 元素，宣告能與 AIR 應用程式產生關聯的檔案類型。

安裝 AIR 應用程式時，會向作業系統登錄所宣告的檔案類型。如果這些檔案類型尚未與其它應用程式產生關聯，就會與 AIR 應用程式產生關聯。若要覆寫檔案類型與其它應用程式之間的現有關聯，請在執行階段使用 NativeApplication.setAsDefaultApplication() 方法 (建議先經過使用者的同意)。

備註：執行階段方法都只能管理應用程式描述器中所宣告檔案類型的關聯。

fileTypes 是選擇性元素。

父元素：第 178 頁「application」

子元素：第 189 頁「fileType」

內容
fileTypes 元素可包含任何數目的 fileType 元素。

範例

```
<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>
```

字體

說明可在 AIR 應用程式中使用的單一自訂字體。

父元素：第 186 頁「embedFonts」

子元素：第 190 頁「fontName」、第 190 頁「fontPath」

內容

指定自訂字體名稱及其路徑的元素。

範例

```
<font>
   <fontPath>ttf/space age.ttf</fontPath>
   <fontName>space age</fontName>
</font>
```

fontName

指定自訂字體的名稱。

父元素：第 190 頁「字體」

子元素：無

內容

要在 StageText.fontFamily 中指定的自訂字體名稱

範例

```
<fontName>space age</fontName>
```

fontPath

提供自訂字體檔案的位置。

父元素：第 190 頁「字體」

子元素：無

內容
自訂字體檔案的路徑 (以來源為基準)。

範例
```
<fontPath>ttf/space age.ttf</fontPath>
```

forceCPURenderModeForDevices

Adobe AIR 3.7 及更新的版本、僅 **iOS —** 選擇性

針對指定的裝置組合強制套用 CPU 顯示模式。這個功能會有效地讓您可以針對其餘 iOS 裝置選擇性地啟用 GPU 顯示模式。

您可以將這個標記新增為 iPhone 標記的子系，並且指定以空格分隔的裝置機型名稱清單。有效的裝置機型名稱包括：

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		

父元素：第 197 頁「iPhone」、第 195 頁「initialWindow」

子元素：無

內容
以空格分隔的裝置機型名稱清單。

特質：
無。

範例
iOS：

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

fullScreen

Adobe AIR 2.0 及更新的版本、**iOS** 與 **Android** — 選擇性

指定應用程式是否要在全螢幕模式下啟動。

父元素：第 195 頁「initialWindow」

子元素：無

內容
true 或 false (預設值)

範例
```
<fullscreen>true</fullscreen>
```

height

Adobe AIR 1.0 及更新的版本 — 選擇性

應用程式主要視窗的初始高度。

如果未設定高度，高度將由根 SWF 檔案中的設定決定。若為 HTML 類型的 AIR 應用程式案例，則由作業系統決定。

在 AIR 2 中，視窗的最大高度從 2048 像素變更為 4096 像素。

父元素：第 195 頁「initialWindow」

子元素：無

內容
最大值為 4095 的正整數。

範例
```
<height>4095</height>
```

icon

Adobe AIR 1.0 及更新的版本 — 選擇性

icon 屬性會指定一個或多個要用於應用程式的圖示檔案。您可以選擇是否要包含圖示。如果未指定 icon 屬性，作業系統便會顯示預設圖示。

指定的路徑必須相對於應用程式根目錄。圖示檔案必須是 PNG 格式。您可以指定下列所有圖示大小：

如果在元素中指定大小，檔案中的影像就必須是指定的大小。如果未提供任何大小，作業系統會自行縮放相近大小的圖示以符合需求。

備註：指定的圖示不會自動加入至 AIR 套件。在封裝應用程式時，您必須將圖示檔案納入這些檔案的正確相對位置。

您可以為每一個可用大小準備一個影像，以獲得最佳的外觀效果。此外，請確保圖示在 16 和 32 位元色彩模式下，都能得到一致的外觀效果。

父元素：第 178 頁「application」

子元素：第 193 頁「imageNxN」

內容
每個所需圖示大小的 imageNxN 元素。

範例
```
<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 及更新的版本 — 必要

應用程式的識別名稱字串，又稱為應用程式 ID。通常會使用保留的 DNS 樣式識別名稱，但不一定要使用此樣式。

父元素：第 178 頁「application」

子元素：無

內容
ID 值只能由下列字元組成：

- 0–9
- a–z
- A–Z
- . (點)
- - (連字符號)

這個值必須介於 1 到 212 個字元之間。這是必要元素。

範例
```
<id>org.example.application</id>
```

imageNxN

Adobe AIR 1.0 及更新的版本 — 選擇性

定義與應用程式目錄相對的圖示路徑。

您可以使用下列圖示影像，每個影像會指定不同的圖示大小：

- 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+)

圖示必須是與影像元素所指定的大小相同的 PNG 圖像。您必須將圖示檔案加入應用程式套件中，應用程式描述器文件中使用的圖示並不會自動加入應用程式套件。

父元素：第 178 頁「application」

子元素：無

內容
圖示的檔案路徑可以包含下列字元以外的任何 Unicode (UTF-8) 字元，多種檔案系統都禁止在檔案名稱中使用下列字元：

字元	十六進位碼
右列十六進位碼範圍內的字元	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

範例

```
<image32x32>icons/icon32.png</image32x32>
```

InfoAdditions

Adobe AIR 1.0 及更新的版本 — 選擇性

允許您指定 iOS 應用程式的其他屬性。

父元素：第 197 頁「iPhone」

子元素：iOS Info.plist 元素

內容
包含子元素，這些子元素會指定索引鍵值配對，做為應用程式的 Info.plist 設定：InfoAdditions 元素的內容應該包括在 CDATA 區塊中。

如需有關索引鍵值配對以及如何以 XML 表示它們的詳細資訊，請參閱「Apple iPhone 參考元件庫」中的資訊屬性清單索引鍵參考。

範例

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

更多說明主題

第 70 頁「iOS 設定」

initialWindow

Adobe AIR 1.0 及更新的版本 — 必要

定義主要內容檔案與初始應用程式外觀。

父元素：第 178 頁「application」

子元素：下列所有元素都可以做為 initialWindow 元素的子系顯示。不過，需視 AIR 是否支援平台上的視窗而定，某些元素會被忽略：

元素	桌上型	行動裝置
第 181 頁「aspectRatio」	忽略	使用
第 181 頁「autoOrients」	忽略	使用
第 183 頁「content」	使用	使用
第 184 頁「depthAndStencil」	使用	使用
第 192 頁「fullScreen」	忽略	使用
第 192 頁「height」	使用	忽略
第 199 頁「maximizable」	使用	忽略
第 199 頁「maxSize」	使用	忽略
第 200 頁「minimizable」	使用	忽略
第 200 頁「minSize」	使用	忽略
第 202 頁「renderMode」	使用 (AIR 3.0 及更新版本)	使用
第 203 頁「requestedDisplayResolution」	使用 (AIR 3.6 及更新版本)	忽略
第 204 頁「resizable」	使用	忽略
第 204 頁「softKeyboardBehavior」	忽略	使用

元素	桌上型	行動裝置
第 206 頁「systemChrome」	使用	忽略
第 207 頁「title」	使用	忽略
第 207 頁「transparent」	使用	忽略
第 208 頁「visible」	使用	忽略
第 209 頁「width」	使用	忽略
第 209 頁「x」	使用	忽略
第 210 頁「y」	使用	忽略

內容
定義應用程式外觀與行為的子元素。

範例

```
<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 及更新的版本 — 選擇性

指出預設安裝目錄的子目錄。

在 Windows 中，預設的安裝子目錄為 Program Files 目錄；在 Mac OS 中則是 /Applications 目錄。在 Linux 中則安裝於 /opt/。舉例來說，如果將 installFolder 屬性設定為 "Acme"，而且應用程式的名稱為 "ExampleApp"，應用程式在 Windows 中的安裝位置就會是 C:\Program Files\Acme\ExampleApp、在 Mac OS 中的安裝位置是 /Applications/Acme/Example.app，而在 Linux 中的安裝位置則是 /opt/Acme/ExampleApp。

installFolder 是選擇性屬性。如果未指定 installFolder 屬性，應用程式便會根據 name 屬性，安裝在預設安裝目錄的子目錄中。

父元素：第 178 頁「application」

子元素：無

內容
除了各種檔案系統都禁止當做資料夾名稱使用的字元以外，installFolder 屬性可以包含任何 Unicode (UTF-8) 字元 (如需例外字元清單，請參閱 filename 屬性)。

如果您希望指定巢狀子目錄，可以使用正斜線 (/) 字元做為目錄分隔符號字元。

範例
```
<installFolder>utilities/toolA</installFolder>
```

iPhone

僅 **Adobe AIR 2.0**、**iOS —** 選擇性

定義 iOS 特定的應用程式屬性。

父元素：第 178 頁 「application」

子元素：

- 第 186 頁 「Entitlements」
- 第 187 頁 「externalSwfs」
- 第 191 頁 「forceCPURenderModeForDevices」
- 第 194 頁 「InfoAdditions」
- 第 203 頁 「requestedDisplayResolution」

更多說明主題
第 70 頁 「iOS 設定」

manifest

Adobe AIR 2.5 及更新的版本、僅 **Android —** 選擇性

指定要為應用程式新增至 Android 資訊清單檔案的資訊。

父元素：第 198 頁 「manifestAdditions」

子元素：由 Android SDK 定義。

內容
技術上而言，manifest 元素不是 AIR 應用程式描述器結構的一部分。它是 Android 資訊清單 XML 文件的根元素。任何在 manifest 元素中放置的內容必須遵循 AndroidManifest.xml 結構。當您使用 AIR 工具產生 APK 檔案時，會將 manifest 元素中的資訊複製到應用程式產生的 AndroidManifest.xml 對應部分。

如果您指定的 Android 資訊清單值只在比 AIR 直接支援版本更新的 SDK 版本中提供，則在封裝應用程式時，您必須將 -platformsdk 旗標設定為 ADT。請將旗標設定為檔案系統路徑，該路徑指向支援所要新增之值的 Android SDK 版本。

manifest 元素本身必須包括在 AIR 應用程式描述器的 CDATA 區塊中。

範例

```
<![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>
]]>
```

更多說明主題

第 65 頁「Android 設定」

AndroidManifest.xml 檔案

manifestAdditions

Adobe AIR 2.5 及更新的版本、僅 **Android**

指定要新增至 Android 資訊清單檔案的資訊。

每個 Android 應用程式都包括定義基本應用程式屬性的資訊清單檔案。Android 資訊清單在概念上類似於 AIR 應用程式描述器。AIR for Android 應用程式同時具有應用程式描述器與自動產生的 Android 資訊清單檔案。封裝 AIR for Android 應用程式時，會將此 manifestAdditions 元素中的資訊新增至對應的 Android 資訊清單文件部分。

父元素：第 177 頁「android」

子元素：第 197 頁「manifest」

內容

在 manifestAdditions 元素中的資訊會新增至 AndroidManifest XML 文件中。

AIR 會在產生的 Android 資訊清單文件中設定數個資訊清單項目，以確保應用程式與執行階段功能可正常運作。您無法覆寫下列設定：

您無法設定 manifest 元素的下列特質：

- package
- android:versionCode
- android:versionName

您無法設定主要 activity 元素的下列特質：

- android:label
- android:icon

您無法設定 application 元素的下列特質：

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

- android:launchMode

範例

```
<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>
```

更多說明主題

第 65 頁 「Android 設定」

AndroidManifest.xml 檔案

maximizable

Adobe AIR 1.0 及更新的版本 — 選擇性

指定視窗是否可最大化。

備註：在 Mac OS X 這類作業系統中，將視窗最大化也是一項調整大小作業，因此 maximizable 和 resizable 都必須設定為 false，以避免視窗大小遭縮放或調整。

父元素：第 195 頁 「initialWindow」

子元素：無

內容
true (預設值) 或 false

範例

```
<maximizable>false</maximizable>
```

maxSize

Adobe AIR 1.0 及更新的版本 — 選擇性

視窗的大小上限。如果您未設定大小上限，則由作業系統決定。

父元素：第 195 頁 「initialWindow」

子元素：無

內容
代表寬度與高度上限的兩個整數，以空格分隔。

備註：在 AIR 2 中，AIR 支援的視窗大小上限從 2048x2048 像素增加至 4096x4096 像素 (因為螢幕座標從零開始，所以您可以使用的寬度或高度上限為 4095)。

範例
```
<maxSize>1024 360</maxSize>
```

minimizable

Adobe AIR 1.0 及更新的版本 — 選擇性

指定視窗是否可最小化。

父元素：第 195 頁「initialWindow」

子元素：無

內容
true (預設值) 或 false

範例
```
<minimizable>false</minimizable>
```

minSize

Adobe AIR 1.0 及更新的版本 — 選擇性

指定視窗允許的大小下限。

父元素：第 195 頁「initialWindow」

子元素：無

內容
代表寬度與高度下限的兩個整數，以空格分隔。請注意，作業系統實行的大小下限優先於應用程式描述器中設定的值。

範例
```
<minSize>120 60</minSize>
```

name

Adobe AIR 1.0 及更新的版本 — 選擇性

AIR 應用程式安裝程式顯示的應用程式標題。

如果未指定 name 元素，AIR 應用程式安裝程式就會顯示 filename 做為應用程式名稱。

父元素：第 178 頁「application」

子元素：第 206 頁「text」

內容
如果您指定單一文字節點 (而非多個 <text> 元素)，則不論系統語言為何，AIR 應用程式安裝程式都會使用這個名稱。

在 AIR 1.0 應用程式描述器結構中，只允許為該名稱 (而非多個 text 元素) 定義一個簡單文字節點。在 AIR 1.1 (或更新版本) 中，您可以在 name 元素內指定多個語言。

如 RFC4646 (http://www.ietf.org/rfc/rfc4646.txt) 中所定義，每一個 text 元素的 xml:lang 特質都可以指定一個語言代碼。

AIR 應用程式安裝程式會使用最符合使用者作業系統之使用者介面語言的名稱。例如，假設正在執行一項安裝作業，其中應用程式描述器檔案的 name 元素包含代表 en (英文) 地區的值。如果作業系統將 en (英文) 識別為使用者介面語言，那麼 AIR 應用程式安裝程式就會使用這個 en 名稱。此外，如果系統使用者介面語言為 en-US (美式英文)，安裝程式也會使用這個 en 名稱。但是，如果使用者介面語言為 en-US，而且應用程式描述器檔案定義了 en-US 和 en-GB 這兩個名稱，那麼 AIR 應用程式安裝程式就會使用 en-US 這個值。如果應用程式定義的名稱都不符合系統使用者介面語言，AIR 應用程式安裝程式就會使用在應用程式描述器檔案中定義的第一個 name 值。

name 元素只會定義在 AIR 應用程式安裝程式中使用的應用程式標題。AIR 應用程式的安裝程式支援下列語言：繁體中文、簡體中文、捷克文、荷蘭文、英文、法文、德文、義大利文、日文、韓文、波蘭文、巴西葡萄牙文、俄文、西班牙文、瑞典文和土耳其文。AIR 應用程式安裝程式會根據系統使用者介面語言選取所顯示的語言 (針對應用程式標題和說明以外的文字而言)。這個語言選取項目與應用程式描述器檔案中的設定無關。

針對已安裝且正在執行中的應用程式，name 元素 「不會」定義可供其使用的地區。如需有關開發多國語言應用程式的詳細資訊，請參閱第 254 頁 「當地語系化 AIR 應用程式」。

範例
下列範例會使用簡單文字節點定義名稱：

```
<name>Test Application</name>
```

下列範例 (適用於 AIR 1.1 及更新的版本) 使用 <text> 元素節點指定三種語言 (英文、法文和西班牙文) 的名稱：

```
<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 及更新的版本 — 必要

識別檔案類型的名稱。

父元素：第 189 頁 「fileType」

子元素：無

內容
表示檔案類型名稱的字串。

範例

```
<name>adobe.VideoFile</name>
```

programMenuFolder

Adobe AIR 1.0 及更新的版本 — 選擇性

指出 Windows 作業系統 「所有程式」功能表或 Linux 「應用程式」選單中的應用程式捷徑位置 (在其他作業系統中，目前會忽略這個設定)。

父元素：第 178 頁 「application」

子元素：無

內容
除了各種檔案系統都禁止當做資料夾名稱使用的字元以外，用於 programMenuFolder 值的字串可以包含任何 Unicode (UTF-8) 字元 (如需例外字元清單，請參閱 filename 元素)。請「勿」使用正斜線 (/) 字元做為這個值的最後一個字元。

範例
```
<programMenuFolder>Example Company/Sample Application</programMenuFolder>
```

publisherID

Adobe AIR 1.5.3 及更新的版本 — 選擇性

識別發行者 ID，以便更新原先使用 AIR 1.5.2 版或較舊版本所建立的 AIR 應用程式。

只有在建立應用程式更新時，才需指定發行者 ID。publisherID 元素的值必須符合 AIR 為先前應用程式版本所產生的發行者 ID。若為已安裝的應用程式，則可在安裝應用程式的資料夾中 (位於 META-INF/AIR/publisherid 檔案中) 找到發行者 ID。

使用 AIR 1.5.3 或更新版本建立的新應用程式不應指定發行者 ID。

如需詳細資訊，請參閱第 162 頁「關於 AIR 發行者識別名稱」。

父元素：第 178 頁「application」

子元素：無

內容
發行者 ID 字串。

範例
```
<publisherID>B146A943FBD637B68C334022D304CEA226D129B4.1</publisherID>
```

renderMode

Adobe AIR 2.0 及更新的版本 — 選擇性

指定是否使用圖像處理單元 (GPU) 加速 (如果目前的電腦裝置支援此功能)。

父元素：第 195 頁「initialWindow」

子元素：無

內容
下列其中一個值：

- auto (預設值) — 目前會退回 CPU 模式。
- cpu — 未使用硬體加速。
- direct — CPU 出現顯示視覺調整，位圖複製使用 GPU。在 AIR 3+ 提供。

  備註：若要在行動平台上使用 AIR 來利用 Flash 內容的 GPU 加速，Adobe 建議您使用 renderMode="direct" (即 Stage3D)，而不要使用 renderMode="gpu"。Adobe 正式支援並建議使用下列 Stage3D 架構：Starling (2D) 和 Away3D (3D)。如需有關 Stage3D 和 Starling/Away3D 的詳細資訊，請參閱 http://gaming.adobe.com/getstarted/。
- gpu — 使用硬體加速 (若有的話)。

  重要事項：請勿為 Flex 應用程式使用 GPU 顯示模式。

範例

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

requestedDisplayResolution

Adobe AIR 2.6 及更新版本、僅 **iOS**；**Adobe AIR 3.6** 及更新版本、**OS X —** 選擇性

指定應用程式要在具有高解析度螢幕的裝置或電腦顯示上使用標準解析度，還是高解析度。設定為預設值 「標準」時，應用程式會將螢幕當成標準解析度螢幕。設定為 「高」時，應用程式可以處理每個高解析度像素。

例如，在 640x960 高解析度 iPhone 螢幕上，如果設定為 「標準」，則全螢幕舞台尺寸為 320x480，並且每一應用程式像素會使用 4 螢幕像素來顯示。如果設定為 「高」，則全螢幕舞台尺寸為 640x960。

在具有標準解析度螢幕的裝置上，不論使用哪個設定，舞台尺寸都會符合螢幕尺寸。

如果 requestedDisplayResolution 元素位於 iPhone 元素的巢狀結構內，則會套用至 iOS 裝置。在此情況下，您可以使用 excludeDevices 特質來指定哪些裝置不套用此設定。

如果 requestedDisplayResolution 元素位於 initialWindow 元素的巢狀結構內，則會套用至支援高解析度顯示器之 MacBook Pro 電腦上的桌上型 AIR 應用程式。指定的值會套用至應用程式中使用的所有原生視窗。AIR 3.6 及更新版本中支援 requestedDisplayResolution 元素以巢狀方式出現在 initialWindow 元素中。

父元素：第 197 頁 「iPhone」、第 195 頁 「initialWindow」

子元素：無

內容

standard (預設值) 或 high。

特質：

excludeDevices — 以空格分隔之 iOS 機型名稱或機型名稱前置詞的清單。這允許開發人員可讓一些裝置使用高解析度，其他則使用標準解析度。只有 iOS 才可以使用這個特質 (requestedDisplayResolution 元素位於 iPhone 元素的巢狀結構內)。excludeDevices 特質可以在 AIR 3.6 及更新版本中取得。

若這個特質中指定了任何裝置的機型名稱，則 requestedDisplayResolution 值與指定的值相反。換句話說，如果 requestedDisplayResolution 值為 「高」，則排除的裝置會使用標準解析度。如果 requestedDisplayResolution 值為 「標準」，則排除的裝置會使用高解析度。

值為 iOS 裝置機型名稱或機型名稱前置詞。例如，值 iPad3,1 特別表示 Wi-Fi 第 3 代 iPad (而不是 GSM 或 CDMA 第 3 代 iPad)。或者，值 iPad3 表示任何第 3 代 iPad。iOS 機型名稱的非官方清單可以在 iPhone wiki 機型頁取得。

範例

桌上型：

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

iOS：

```
<iPhone>
    <requestedDisplayResolution excludeDevices="iPad3 iPad4">high</requestedDisplayResolution>
</iPhone>
```

resizable

Adobe AIR 1.0 及更新的版本 — 選擇性

指定視窗是否以調整大小。

備註：在 Mac OS X 這類作業系統中，將視窗最大化也是一項調整大小作業，因此 maximizable 和 resizable 都必須設定為 false，以避免視窗大小遭縮放或調整。

父元素：第 195 頁 「initialWindow」

子元素：

內容
true (預設值) 或 false

範例
```
<resizable>false</resizable>
```

softKeyboardBehavior

Adobe AIR 2.6 及更新的版本、行動描述檔 — 選擇性

指定顯示虛擬鍵盤時的應用程式預設行為。預設行為是向上平移應用程式。執行階段會在螢幕上保留取得焦點的文字欄位或互動式物件。如果您的應用程式沒有提供自己的鍵盤處理邏輯，請使用 pan 選項。

您也可以將 softKeyboardBehavior 元素設定為 none，以關閉自動行為。在此例中，文字欄位與互動式物件會在軟體鍵盤升起顯示時傳送 SoftKeyboardEvent，但是執行階段不會平移應用程式或調整其大小。您的應用程式必須將文字輸入區域保持在畫面中。

父元素：第 195 頁 「initialWindow」

子元素：無

內容
none 或 pan。預設值為 pan。

範例
```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

更多說明主題
SoftKeyboardEvent

supportedLanguages

Adobe AIR 3.2 及更新的版本 - 選擇性

識別應用程式支援的語言。這個元素僅限 iOS、Mac 固定執行階段和 Android 應用程式使用。所有其他應用程式類型則會忽略這個元素。

如果您未指定這個元素，封裝程式預設會根據應用程式類型執行下列動作：

- iOS — AIR 執行階段支援的所有語言會在 iOS App Store 中列為應用程式的支援語言。
- Mac 固定執行階段 — 以固定組合包封裝的應用程式沒有當地語系化資訊。

- Android — 應用程式組合包具有 AIR 執行階段支援之所有語言的資源。

父元素：第 178 頁 「application」

子元素：無

內容
以空格分隔的支援語言清單。有效的語言值為 AIR 執行階段支援之語言的 ISO 639-1 值：en、de、es、fr、it、ja、ko、pt、ru、cs、nl、pl、sv、tr、zh、da、nb、iw。

封裝程式會因為 <supportedLanguages> 元素的空白值而產生錯誤。

備註：如果您使用 <supportedLanguages> 標籤且其中未包含語言，則當地語系化標籤 (例如名稱標籤) 會忽略該語言的值。如果原生擴充功能具有並非由 <supportedLangauges> 標籤指定之語言的資源，則會發出警告並忽略該語言的資源。

範例
```
<supportedLanguages>en ja fr es</supportedLanguages>
```

supportedProfiles

Adobe AIR 2.0 及更新的版本 — 選擇性

識別應用程式支援的描述檔。

父元素：第 178 頁 「application」

子元素：無

內容
您可以在 supportedProfiles 元素中包含下列任一值：

- desktop：桌上型描述檔適用於使用 AIR 檔案安裝在桌上型電腦的 AIR 應用程式。這些應用程式沒有 NativeProcess 類別 (可提供和原生應用程式通訊功能) 的存取權。
- extendedDesktop：延伸桌上型描述檔適用於使用原生應用程式安裝程式安裝在桌上型電腦的 AIR 應用程式。這些應用程式具有 NativeProcess 類別 (可提供和原生應用程式通訊功能) 的存取權。
- mobileDevice— 行動裝置描述檔適用於行動應用程式。
- extendedMobileDevice：目前尚無法使用延伸行動裝置描述檔。

supportedProfiles 是選擇性屬性。若您未在應用程式描述器檔案中包含這個元素時，可以針對任何描述檔編譯和部署應用程式。

若要指定多重描述檔，請以空格字元分隔每一個描述檔。例如，下列設定指定應用程式只能用於 desktop 和 extended 描述檔中。

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

備註：當您使用 ADL 執行應用程式且沒有指定 ADL -profile 選項值時，會使用應用程式描述器中的第一個描述檔 (如果在應用程式描述器中也沒有指定任何描述檔，則會使用 desktop 描述檔)。

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

更多說明主題

第 211 頁 「裝置描述檔」

第 65 頁 「支援的描述檔」

systemChrome

Adobe AIR 1.0 及更新的版本 — 選擇性

指定是否使用作業系統提供的標準標題列、邊界及控制項，來建立初始應用程式視窗。

您無法在執行階段變更視窗的系統顏色設定。

父元素：第 195 頁 「initialWindow」

子元素：無

內容
下列其中一個值：

- none — 不提供系統顏色。應用程式 (或是 Flex 等應用程式架構) 必須負責顯示視窗顏色。
- standard (預設值) — 由作業系統提供系統顏色。

範例

```
<systemChrome>standard</systemChrome>
```

text

Adobe AIR 1.1 及更新的版本 — 選擇性

指定當地語系化字串。

如 http://www.ietf.org/rfc/rfc4646.txt (http://www.ietf.org/rfc/rfc4646.txt) 中所定義，text 元素的 xml:lang 特質可以指定一個語言代碼。

AIR 應用程式安裝程式會將 text 元素與最符合使用者作業系統之使用者介面語言的 xml:lang 特質值搭配使用。

例如，當安裝的 text 元素包含 en (英文) 地區設定值時。如果作業系統將 en (英文) 識別為使用者介面語言，那麼 AIR 應用程式安裝程式就會使用這個 en 名稱。此外，如果系統使用者介面語言為 en-US (美式英文)，安裝程式也會使用這個 en 名稱。但是，如果使用者介面語言為 en-US，而且應用程式描述器檔案定義了 en-US 和 en-GB 這兩個名稱，那麼 AIR 應用程式安裝程式就會使用 en-US 這個值。

如果應用程式定義的 text 元素都不符合系統使用者介面語言，AIR 應用程式安裝程式就會使用在應用程式描述器檔案中定義的第一個 name 值。

父元素：

- 第 200 頁 「name」
- 第 185 頁 「description」

子元素：無

內容
指定地區設定的 xml:lang 特質以及當地語系化文字的字串。

範例

```
<text xml:lang="fr">Bonjour AIR</text>
```

title

Adobe AIR 1.0 及更新的版本 — 選擇性

指定在初始應用程式視窗標題列中顯示的標題。

只有將 systemChrome 元素設定為 standard 時，才會顯示標題。

父元素：第 195 頁 「initialWindow」

子元素：無

內容
包含視窗標題的字串。

範例

```
<title>Example Window Title</title>
```

transparent

Adobe AIR 1.0 及更新的版本 — 選擇性

指定初始應用程式視窗是否與桌面進行 Alpha 混合。

啟用套用透明度的視窗時，速度會變慢，並且會耗用更多記憶體。您無法在執行階段變更透明度設定。

重要事項：當 systemChrome 設定為 none 時，您只能將 transparent 設定為 true。

父元素：第 195 頁 「initialWindow」

子元素：無

內容
true 或 false (預設值)

範例

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

version

Adobe AIR 1.0 到 **2.0** — 必要；在 **AIR 2.5** 及更新的版本中不允許

指定應用程式的版本資訊

版本字串是應用程式定義的指示項。AIR 不會以任何方式解譯版本字串。因此，版本 “3.0” 不會被認定為比版本 “2.0” 還要新。範例："1.0"、".4"、"0.5"、"4.9"、"1.3.4a"。

在 AIR 2.5 及更新的版本中，versionNumber 與 versionLabel 元素已取代 version 元素。

父元素：第 178 頁 「application」

子元素：無

內容
包含應用程式版本的字串。

範例

```
<version>0.1 Alpha</version>
```

versionLabel

Adobe AIR 2.5 及更新的版本 — 選擇性

指定使用者易懂的版本字串。

安裝對話方塊中會顯示版本標籤值，而不是 versionNumber 元素的值。如果未使用 versionLabel，則 versionNumber 會同時用於兩者。

父元素：第 178 頁「application」

子元素：無

內容

包含公開顯示的版本文字字串。

範例

```
<versionLabel>0.9 Beta</versionlabel>
```

versionNumber

Adobe AIR 2.5 及更新的版本 — 必要

應用程式版本號碼。

父元素：第 178 頁「application」

子元素：無

內容

版本號碼可包含多達三個以句號分隔的整數序列。每個整數必須是 0 到 999 之間的數字 (含)。

範例

```
<versionNumber>1.0.657</versionNumber>

<versionNumber>10</versionNumber>

<versionNumber>0.01</versionNumber>
```

visible

Adobe AIR 1.0 及更新的版本 — 選擇性

指定初始應用程式視窗在建立後是否立即可見。

AIR 視窗 (包括初始視窗) 預設會在隱藏狀態下建立。您可以呼叫 NativeWindow 物件的 activate() 方法，或是將 visible 屬性設定為 true 以顯示視窗。您可能會希望一開始先讓主要視窗呈現隱藏狀態，如此便不會顯示視窗位置、視窗大小，以及視窗內容配置的變更。

除非在 MXML 定義中將 visible 特質設定為 false，否則 Flex mx:WindowedApplication 組件會在傳送 applicationComplete 事件之前，自動顯示並立即啟動視窗。

行動描述檔 (不支援視窗) 中的裝置會忽略可見設定。

父元素：第 195 頁 「initialWindow」

子元素：無

內容
true 或 false (預設值)

範例
```
<visible>true</visible>
```

width

Adobe AIR 1.0 及更新的版本 — 選擇性

應用程式主要視窗的初始寬度。

如果未設定寬度，寬度將由根 SWF 檔案中的設定決定。若為 HTML 類型的 AIR 應用程式案例，則由作業系統決定。

在 AIR 2 中，視窗的最大寬度從 2048 像素變更為 4096 像素。

父元素：第 195 頁 「initialWindow」

子元素：無

內容
最大值為 4095 的正整數。

範例
```
<width>1024</width>
```

x

Adobe AIR 1.0 及更新的版本 — 選擇性

初始應用程式視窗的水平位置。

在大部分的情況下，最好是讓作業系統決定視窗的初始位置，而不是指定固定值。

螢幕座標系統的原點 (0,0) 為主要桌面螢幕的左上角 (由作業系統決定)。

父元素：第 195 頁 「initialWindow」

子元素：無

內容
整數值。

範例
```
<x>120</x>
```

y

Adobe AIR 1.0 及更新的版本 — 選擇性

初始應用程式視窗的垂直位置。

在大部分的情況下，最好是讓作業系統決定視窗的初始位置，而不是指定固定值。

螢幕座標系統的原點 (0,0) 為主要桌面螢幕的左上角 (由作業系統決定)。

父元素：第 195 頁「initialWindow」

子元素：無

內容
整數值。

範例

```
<y>250</y>
```

第 15 章 裝置描述檔

Adobe AIR 2 以及更新的版本

描述檔是一種機制，用以定義應用程式運作的計算裝置類別。描述檔會定義一組特定裝置類別通常支援的 API 與功能。可用的描述檔包括：

- desktop
- extendedDesktop
- mobileDevice
- extendedMobileDevice

您可以在應用程式描述器中定義您的應用程式描述檔。使用描述檔所述之電腦與裝置的使用者可以安裝您的應用程式，而使用其他電腦與裝置的使用者則無法安裝。例如，如果您在應用程式描述器中只包括桌上型描述檔，則使用者只能在桌上型電腦上安裝和執行您的應用程式。

如果您的應用程式無法真正支援您使用的描述檔，則在這樣環境下的使用者經驗可能會不佳。如果未在應用程式描述器中指定任何描述檔，則 AIR 不會限制您的應用程式。您可以使用任何支援的格式來封裝應用程式，而使用任何描述檔的裝置使用者都可以安裝它，不過，應用程式有可能在執行階段無法正常運作。

因此當您封裝應用程式時，請盡可能強制描述檔限制。例如，如果您只包括 extendedDesktop 描述檔，您便無法將應用程式封裝成 AIR 檔案，只能封裝成原生安裝程式。同樣地，如果您未包括 mobileDevice 描述檔，則無法將應用程式封裝為 Android APK。

單一電腦裝置可以支援多個描述檔。例如，桌上型電腦上的 AIR 同時支援 desktop 描述檔與 extendedDesktop 描述檔中的應用程式。不過，延伸桌上型描述檔應用程式可以和原生處理程序通訊，而且「必須」封裝為原生安裝程式 (exe、dmg、deb 或 rpm)。另一方面，桌上型描述檔應用程式無法與原生處理程序通訊。桌上型描述檔應用程式可以封裝成 AIR 檔案或原生安裝程式。

包括在描述檔中的功能表示描述檔中所定義的裝置類別通常可支援該功能。不過，並不表示描述檔中的每個裝置皆可支援每項功能。例如，大部分但並非所有的行動電話都包含加速計。不普遍支援的類別與功能通常具有 Boolean 屬性，您可以在使用該功能前進行檢查。例如，以加速計為例，您可以測試靜態屬性 Accelerometer.isSupported，以判斷目前的裝置是否具有支援的加速計。

您可以使用應用程式描述器中的 supportedProfiles 元素，將下列描述檔指定給 AIR 應用程式：

桌上型 桌上型描述檔會針對在桌上型電腦中安裝為 AIR 檔案的 AIR 應用程式，定義一組功能。這些應用程式會在支援的桌上型平台 (Mac OS、Windows 以及 Linux) 上安裝和執行。在 AIR 2 版本之前開發的 AIR 應用程式都可視為使用桌上型描述檔。有些 API 在此描述檔中無法運作。例如，桌面應用程式無法與原生處理程序通訊。

延伸桌上型 延伸桌上型描述檔會針對以原生安裝程式封裝和安裝的 AIR 應用程式，定義一組功能。這些原生安裝程式在 Windows 上為 EXE 檔，在 Mac OS 上為 DMG 檔，而在 Linux 上則為 BIN、DEB 或 RPM 檔。延伸桌上型應用程式具有桌上型描述檔應用程式所沒有的額外功能。如需詳細資訊，請參閱第 49 頁 「封裝桌上型原生安裝程式」。

行動裝置 行動裝置描述檔會針對行動裝置 (如行動電話和數位板) 上安裝的應用程式定義一組功能。這些應用程式會安裝並執行於支援的行動平台上，其中包括 Android、Blackberry Tablet OS 和 iOS。

延伸行動裝置 延伸行動裝置描述檔會針對行動裝置上安裝的應用程式定義一組延伸功能。但目前並無任何裝置支援此描述檔。

在應用程式描述器檔案中限制目標描述檔

Adobe AIR 2 以及更新的版本

從 AIR 2 開始，應用程式描述器檔案包含一個 supportedProfiles 元素，此元素可讓您限制目標描述檔。例如，下列設定指定應用程式只能使用桌上型描述檔：

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

設定此元素時，應用程式只能以您列出的描述檔來封裝。請使用下列的值：

- desktop：桌上型描述檔
- extendedDesktop：延伸桌上型描述檔
- mobileDevice：行動裝置描述檔

supportedProfiles 是選擇性元素。若您未在應用程式描述器檔案中包含這個元素，則您可以使用任何描述檔來封裝和部署應用程式。

若要在 supportedProfiles 元素指定多個描述檔，請使用空格字元分隔每個檔案，如下所示：

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

不同描述檔的功能

Adobe AIR 2 以及更新的版本

下表列出所有描述檔均不支援的類別與功能。

類別或功能	desktop	extendedDesktop	mobileDevice
Accelerometer (Accelerometer.isSupported)	否	否	檢查
Accessibility (Capabilities.hasAccessibility)	是	是	否
原音回音消除 (Microphone.getEnhancedMicrophone())	是	是	否
ActionScript 2	是	是	否
CacheAsBitmap matrix	否	否	是
Camera (Camera.isSupported)	是	是	是
CameraRoll	否	否	是
CameraUI (CameraUI.isSupported)	否	否	是
固定執行階段組合包	是	是	是
ContextMenu (ContextMenu.isSupported)	是	是	否
DatagramSocket (DatagramSocket.isSupported)	是	是	是
DockIcon (NativeApplication.supportsDockIcon)	檢查	檢查	否

類別或功能	desktop	extendedDesktop	mobileDevice
Drag-and-drop (NativeDragManager.isSupported)	是	是	檢查
EncyptedLocalStore (EncyptedLocalStore.isSupported)	是	是	是
Flash Access (DRMManager.isSupported)	是	是	否
GameInput (GameInput.isSupported)	否	否	否
Geolocation (Geolocation.isSupported)	否	否	檢查
HTMLLoader (HTMLLoader.isSupported)	是	是	否
IME (IME.isSupported)	是	是	檢查
LocalConnection (LocalConnection.isSupported)	是	是	否
Microphone (Microphone.isSupported)	是	是	檢查
多頻道音效 (Capabilities.hasMultiChannelAudio())	否	否	否
原生擴充功能	否	是	是
NativeMenu (NativeMenu.isSupported)	是	是	否
NativeProcess (NativeProcess.isSupported)	否	是	否
NativeWindow (NativeWindow.isSupported)	是	是	否
NetworkInfo (NetworkInfo.isSupported)	是	是	檢查
以預設應用程式開啟檔案	受限	是	否
PrintJob (PrintJob.isSupported	是	是	否
SecureSocket (SecureSocket.isSupported)	是	是	檢查
ServerSocket (ServerSocket.isSupported)	是	是	是
Shader	是	是	受限
Stage3D (Stage.stage3Ds.length)	是	是	是
舞台方向 (Stage.supportsOrientationChange)	否	否	是
StageVideo	否	否	檢查
StageWebView (StageWebView.isSupported)	是	是	是
登入時啟動應用程式 (NativeApplication.supportsStartAtLogin)	是	是	否
StorageVolumeInfo (StorageVolumeInfo.isSupported)	是	是	否
系統閒置模式	否	否	是

類別或功能	desktop	extendedDesktop	mobileDevice
SystemTrayIcon (NativeApplication.supportsSystemTrayIcon)	檢查	檢查	否
Text Layout Framework 輸入	是	是	否
Updater (Updater.isSupported)	是	否	否
XMLSignatureValidator (XMLSignatureValidator.isSupported)	是	是	否

表格中的項目意義如下：

- 「檢查」 — 描述檔中的某些裝置 (但並非所有裝置) 支援這項功能。您應該在使用之前於執行階段檢查是否支援此功能。
- 「受限」 — 支援此功能，但是有許多限制。如需詳細資訊，請參閱相關的文件。
- 「否」 — 描述檔不支援此功能。
- 「是」 — 描述檔支援此功能。請注意，個別的計算裝置可能會缺少某項功能所需的硬體。例如，並非所有的電話都有相機。

以 ADL 除錯時指定描述檔

Adobe AIR 2 以及更新的版本

ADL 會檢查您是否在應用程式描述器檔案的 supportedProfiles 元素中指定支援的描述檔。如有指定，ADL 預設會在除錯時使用列出的第一個支援描述檔。

您可以使用 -profile 命令列引數來指定 ADL 除錯工作階段的描述檔 (請參閱 第 137 頁「AIR Debug Launcher (ADL)」) 不論是否已在應用程式描述器檔案的 supportedProfiles 元素中指定描述檔，您都可以使用此引數。不過，如果您已指定 supportedProfiles 元素，則該元素必須包含您在命令列指定的描述檔。否則，ADL 會產生錯誤。

第 16 章 AIR.SWF 內部瀏覽器 API

自訂隱藏安裝 badge.swf 檔案

除了使用 SDK 隨附的 badge.swf 檔案，您也能自行建立 SWF 檔以供瀏覽器網頁使用。您自訂的 SWF 檔可以透過下列方式與執行階段互動：

- 安裝 AIR 應用程式。請參閱第 219 頁 「從瀏覽器安裝 AIR 應用程式」。
- 檢查是否已安裝特定的 AIR 應用程式。請參閱第 219 頁 「從網頁檢查是否已安裝 AIR 應用程式」。
- 檢查是否已安裝執行階段。請參閱第 218 頁 「檢查是否已安裝執行階段」。
- 啟動使用者系統上已安裝的 AIR 應用程式。請參閱第 220 頁 「從瀏覽器啟動已安裝的 AIR 應用程式」。

這些功能是經由呼叫裝載於 adobe.com 上的 SWF 檔 air.swf 中的 API 所取得。您可以自訂 badge.swf 檔案，並從您自己的 SWF 檔案呼叫 air.swf API。

此外，在瀏覽器內執行的 SWF 檔還能利用 LocalConnection 類別，與執行中的 AIR 應用程式進行通訊。如需詳細資訊，請參閱與其他 Flash Player 和 AIR 實體進行通訊 (適用於 ActionScript 開發人員) 或 Communicating with other Flash Player and AIR instances (適用於 HTML 開發人員)。

重要事項：若要使用本節所述的功能 (以及 air.swf 檔案中的 API)，使用者於 Windows 或 Mac OS 的網頁瀏覽器就必須安裝 Adobe® Flash® Player 9 更新 3 或更新版本。若要在 Linux 上使用隱藏安裝功能，就必須安裝 Flash Player 10 (版本 10,0,12,36 或更新版本)。您可以撰寫程式碼來檢查已安裝的 Flash Player 版本，並於使用者未安裝必要的 Flash Player 版本時提供替代介面予以告知。例如，若是安裝了舊版的 Flash Player，便可以提供下載版 AIR 檔的連結 (而不是使用 badge.swf 檔案或 air.swf API 來安裝應用程式)。

使用 badge.swf 檔案安裝 AIR 應用程式

AIR SDK 和 Flex SDK 隨附有 badge.swf 檔，讓您得以輕鬆使用隱藏安裝功能。badge.swf 能夠從網頁上的連結來安裝執行階段和 AIR 應用程式。所附的項目包括 badge.swf 檔案及其原始碼，可供您的網站進行散佈。

將 **badge.swf** 檔案內嵌在網頁中

1 在 AIR SDK 或 Flex SDK 的 samples/badge 目錄中找出下列檔案，然後將這些檔案加至您的網站伺服器。
   - badge.swf
   - default_badge.html
   - AC_RunActiveContent.js
2 使用文字編輯器開啟 default_badge.html 網頁。
3 在 default_badge.html 網頁中，調整 AC_FL_RunContent() JavaScript 函數內下列項目的 FlashVars 參數定義：

參數	說明
appname	應用程式的名稱，當執行階段未安裝時將由 SWF 檔加以顯示。
appurl	(必要項)。要下載的 AIR 檔所在 URL。必須使用絕對 URL 而非相對 URL。
airversion	(必要項)。若是 1.0 版的執行階段，此參數即設為 1.0。

參數	說明
imageurl	標誌上顯示的影像 (選擇性) 所在 URL。
buttoncolor	下載按鈕的顏色 (指定成十六進位值，例如 FFCC00)。
messagecolor	當執行階段未安裝時，按鈕下方所顯示文字訊息的顏色 (指定成十六進位值，例如 FFCC00)。

4 badge.swf 檔案的最小尺寸為 217 像素寬 x 180 像素高。請將 AC_FL_RunContent() 函數的 width 和 height 參數調整成您所需要的值。

5 重新命名 default_badge.html 檔案並修改其程式碼 (或將之加入其它 HTML 網頁) 以符合您的需求。

備註：對於可載入 badge.swf 檔的 HTML embed 標籤，請勿設定 wmode 特質，請保留預設的設定 ("window")。其他 wmode 設定會使安裝作業無法在某些系統上進行。此外，使用其他 wmode 設定會發生錯誤：“ 錯誤 #2044：未處理的 ErrorEvent:。text=Error #2074：舞台太小，無法符合下載 UI 的大小。”

您也可以編輯並且重新編譯 badge.swf 檔案。如需詳細資訊，請參閱第 216 頁 「修改 badge.swf 檔案」。

從網頁上的隱藏安裝連結安裝 AIR 應用程式

一旦您將隱藏安裝連結加入網頁後，使用者就可以按一下 SWF 檔中的連結來安裝 AIR 應用程式。

1 在已安裝 Flash Player (若使用 Windows 和 Mac OS，則安裝第 9 版更新 3 或更新版本；若使用 Linux，則安裝第 10 版) 的網頁瀏覽器中，瀏覽至 HTML 頁面。

2 按一下網頁上 badge.swf 檔案中的連結。

- 如果您已安裝執行階段，請跳到下一個步驟。
- 如果您尚未安裝執行階段，便會顯示對話方塊詢問您是否要安裝執行階段。請安裝執行階段 (請參閱：第 2 頁 「Adobe AIR 安裝」)，然後繼續進行下一個步驟。

3 在安裝視窗中，保持選取各項預設設定，然後按一下 「繼續」。

在 Windows 電腦上，AIR 會自動執行下列事項：

- 將應用程式安裝於 c:\Program Files\ 中
- 建立應用程式的桌面捷徑
- 建立 「開始」選單捷徑
- 在 「控制台」的 「新增 / 移除程式」中加入應用程式項目

在 Mac OS 上，安裝程式會將應用程式加入至 Applications 目錄 (例如 Mac OS 的 /Applications 目錄)。

在 Linux 電腦上，AIR 會自動執行下列事項：

- 將應用程式安裝至 /opt。
- 建立應用程式的桌面捷徑
- 建立 「開始」選單捷徑
- 在系統套件管理員中加入應用程式項目

4 選取您想要的各種選項，然後按 「安裝」按鈕。

5 安裝完成後，按一下 「完成」。

修改 badge.swf 檔案

Flex SDK 和 AIR SDK 提供 badge.swf 檔案的來源檔案。這些檔案位於 SDK 的 samples/badge 資料夾中：

原始檔案	說明
badge.fla	用來編譯 badge.swf 檔案的原始 Flash 檔案。badge.fla 檔案將編譯成 SWF 9 檔案 (可載入至 Flash Player)。
AIRBadge.as	此 ActionScript 3.0 類別定義了 basdge.fla 檔案中所使用的基底類別。

您可以使用 Flash Professional 重新設計 badge.fla 檔案的視覺介面。

AIRBadge 類別中定義的 AIRBadge() 建構函數會載入裝載於 http://airdownload.adobe.com/air/browserapi/air.swf 的 air.swf 檔案。air.swf 檔案包含了使用隱藏安裝功能的程式碼。

一旦順利載入 air.swf 檔案，便會叫用 AIRBadge 類別的 onInit() 方法：

```
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;
    }
}
```

這段程式碼會將 _air 全域變數設定為所載入 air.swf 檔案的主要類別。此類別具有下列公用方法，供 badge.swf 檔案進行存取以便呼叫隱藏安裝功能：

方法	說明
getStatus()	判斷電腦是否已安裝 (或能否安裝) 執行階段。如需詳細資訊，請參閱第 218 頁「檢查是否已安裝執行階段」。 • runtimeVersion：此字串指定即將安裝的應用程式所需要的執行階段版本 (例如 "1.0.M6")。
installApplication()	在使用者的電腦上安裝指定的應用程式。如需詳細資訊，請參閱第 219 頁「從瀏覽器安裝 AIR 應用程式」。 • url：此字串定義 URL。必須使用絕對 URL 而非相對 URL 路徑。 • runtimeVersion：此字串指定即將安裝的應用程式所需要的執行階段版本 (例如 "2.5")。 • arguments：這些引數會傳遞給安裝後即啟動的應用程式。如果應用程式描述器檔案中的 allowBrowserInvocation 元素設定為 true，應用程式將於安裝後即啟動 (如需有關應用程式描述器檔案的詳細資訊，請參閱 第 173 頁「AIR 應用程式描述器檔案」)。若是以隱藏安裝的方式從瀏覽器啟動應用程式 (由於使用者選擇在安裝後即啟動)，但卻未傳遞這些引數，應用程式的 NativeApplication 物件就不會傳送 BrowserInvokeEvent 物件。傳遞資料給應用程式時，請務必考量安全性方面的隱憂。如需詳細資訊，請參閱第 220 頁「從瀏覽器啟動已安裝的 AIR 應用程式」。

url 和 runtimeVersion 的設定值是透過容器 HTML 網頁中的 FlashVars 設定而傳入 SWF 檔。

如果應用程式將於安裝後自動啟動，您就可以利用 LocalConnection 通訊方式，讓已安裝的應用程式一經叫用便與 badge.swf 檔案聯繫。如需詳細資訊，請參閱與其他 Flash Player 和 AIR 實體進行通訊 (適用於 ActionScript 開發人員) 或 Communicating with other Flash Player and AIR instances (適用於 HTML 開發人員)。

您也可以呼叫 air.swf 檔案的 getApplicationVersion() 方法，檢查是否安裝了特定的應用程式。不論是在應用程式安裝程序進行之前或開始安裝後，皆可呼叫此方法。如需詳細資訊，請參閱第 219 頁 「從網頁檢查是否已安裝 AIR 應用程式」。

載入 air.swf 檔案

您可以取用 air.swf 檔案中的 API 來自行建立 SWF 檔，然後透過瀏覽器內的網頁與執行階段及 AIR 應用程式進行互動。air.swf 檔案裝載於 http://airdownload.adobe.com/air/browserapi/air.swf。若要從您的 SWF 檔中參考 air.swf API，請將 air.swf 檔案載入到與您的 SWF 檔相同的應用程式網域。下列程式碼會示範如何將 air.swf 檔案載入到載入端 SWF 檔所在的應用程式網域：

```
 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;
}
```

一旦載入 air.swf 檔案之後 (此時 Loader 物件的 contentLoaderInfo 物件會傳送 init 事件)，您就可以如下列各節所述，呼叫任何的 air.swf API。

備註：AIR SDK 和 Flex SDK 隨附的 badge.swf 檔會自動載入 air.swf 檔案。請參閱第 215 頁 「使用 badge.swf 檔案安裝 AIR 應用程式」。您可以參照該章節的指示，自行建立 SWF 檔並將 air.swf 檔案載入至其中。

檢查是否已安裝執行階段

SWF 檔可藉由呼叫 air.swf 檔案 (載入來源為 http://airdownload.adobe.com/air/browserapi/air.swf) 所提供的 getStatus() 方法，檢查是否已經安裝執行階段。如需詳細資訊，請參閱第 218 頁 「載入 air.swf 檔案」。

一旦載入 air.swf 檔案之後，SWF 檔即可呼叫 air.swf 檔案的 getStatus() 方法，如下所示：

```
 var status:String = airSWF.getStatus();
```

getStatus() 方法會傳回下列任一字串值，視電腦上的執行階段狀態而定：

字串值	說明
"available"	這部電腦可以安裝執行階段，但目前尚未安裝。
"unavailable"	這部電腦無法安裝執行階段。
"installed"	這部電腦已經安裝執行階段。

如果瀏覽器並未安裝必要的 Flash Player 版本 (若使用 Windows 和 Mac OS，則安裝第 9 版更新 3 或更新版本；若使用 Linux，則安裝第 10 版)，getStatus() 方法就會擲出錯誤。

從網頁檢查是否已安裝 AIR 應用程式

SWF 檔可藉由呼叫 air.swf 檔案 (載入來源為 http://airdownload.adobe.com/air/browserapi/air.swf) 所提供的 getApplicationVersion() 方法，透過比對應用程式 ID 和發行者 ID 的方式，檢查是否已經安裝特定的 AIR 應用程式。如需詳細資訊，請參閱第 218 頁 「載入 air.swf 檔案」。

一旦載入 air.swf 檔案之後，SWF 檔即可呼叫 air.swf 檔案的 getApplicationVersion() 方法，如下所示：

```
 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() 方法具有下列參數：

參數	說明
appID	應用程式的應用程式 ID。如需詳細資訊，請參閱 第 193 頁 「id」。
pubID	應用程式的發行者 ID。如需詳細資訊，請參閱 第 202 頁 「publisherID」。如果提到的應用程式沒有發行者 ID，請將 pubID 參數設定成空字串 ("")。
callback	回呼函數，當做處理常式函數使用。getApplicationVersion() 方法是以非同步方式運作，一旦偵測到已安裝的版本 (或是找不到已安裝的版本) 便會叫用此回呼方法。此回呼方法的定義必須包含單一字串參數，且其值設定為已安裝的應用程式版本字串。如果應用程式尚未安裝，則會傳遞 null 值給此函數，如上述程式碼樣本所示。

如果瀏覽器並未安裝必要的 Flash Player 版本 (若使用 Windows 和 Mac OS，則安裝第 9 版更新 3 或更新版本；若使用 Linux，則安裝第 10 版)，getApplicationVersion() 方法就會擲出錯誤。

備註：至於 AIR 1.5.3，不建議使用發行者 ID。發行者 ID 將不再自動指派給應用程式。為了回溯相容性，應用程式可以繼續指定發行者 ID。

從瀏覽器安裝 AIR 應用程式

SWF 檔可藉由呼叫 air.swf 檔案 (載入來源為 http://airdownload.adobe.com/air/browserapi/air.swf) 所提供的 installApplication() 方法，安裝 AIR 應用程式。如需詳細資訊，請參閱第 218 頁 「載入 air.swf 檔案」。

一旦載入 air.swf 檔案之後，SWF 檔即可呼叫 air.swf 檔案的 installApplication() 方法，如下列程式碼所示：

```
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() 方法會在使用者的電腦上安裝指定的應用程式。這個方法具有下列參數：

參數	說明
url	此字串定義即將安裝的 AIR 檔所在 URL。必須使用絕對 URL 而非相對 URL 路徑。
runtimeVersion	此字串指定即將安裝的應用程式所需要的執行階段版本 (例如 "1.0")。
arguments	此引數陣列會傳遞給安裝後即啟動的應用程式。引數中只能辨識英數字母字元。如需傳遞其它值，請考慮使用編碼資料結構。 如果應用程式描述器檔案中的 allowBrowserInvocation 元素設定為 true，應用程式將於安裝後即啟動 (如需有關應用程式描述器檔案的詳細資訊，請參閱 第 173 頁 「AIR 應用程式描述器檔案」)。若是以隱藏安裝的方式從瀏覽器啟動應用程式 (由於使用者選擇在安裝後即啟動)，但卻未傳遞這些引數，應用程式的 NativeApplication 物件就不會傳送 BrowserInvokeEvent 物件。如需詳細資訊，請參閱第 220 頁 「從瀏覽器啟動已安裝的 AIR 應用程式」。

您必須在使用者事件 (例如按下滑鼠) 的事件處理常式中呼叫 installApplication() 方法，否則此方法不能運作。

如果瀏覽器並未安裝必要的 Flash Player 版本 (若使用 Windows 和 Mac OS，則安裝第 9 版更新 3 或更新版本；若使用 Linux，則安裝第 10 版)，installApplication() 方法就會擲出錯誤。

若要在 Mac OS 上安裝更新版的應用程式，使用者必須具有適當的系統權限，才能在應用程式目錄內進行安裝 (如果應用程式要更新執行階段，則必須具有系統管理權限)。在 Windows 上，使用者必須具有系統管理權限。

您也可以呼叫 air.swf 檔案的 getApplicationVersion() 方法，檢查是否安裝了特定的應用程式。不論是在應用程式安裝程序進行之前或開始安裝後，皆可呼叫此方法。如需詳細資訊，請參閱第 219 頁 「從網頁檢查是否已安裝 AIR 應用程式」。執行中的應用程式可以使用 LocalConnection 類別，與瀏覽器內的 SWF 內容進行通訊。如需詳細資訊，請參閱與其他 Flash Player 和 AIR 實體進行通訊 (適用於 ActionScript 開發人員) 或 Communicating with other Flash Player and AIR instances (適用於 HTML 開發人員)。

從瀏覽器啟動已安裝的 AIR 應用程式

若要使用瀏覽器叫用功能 (以便可從瀏覽器啟動應用程式)，目標應用程式的應用程式描述器檔案必須包含下列設定：

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

如需有關應用程式描述器檔案的詳細資訊，請參閱 第 173 頁 「AIR 應用程式描述器檔案」。

瀏覽器內的 SWF 檔可藉由呼叫 air.swf 檔案 (載入來源為 http://airdownload.adobe.com/air/browserapi/air.swf) 所提供的 launchApplication() 方法，啟動 AIR 應用程式。如需詳細資訊，請參閱第 218 頁 「載入 air.swf 檔案」。

一旦載入 air.swf 檔案之後，SWF 檔即可呼叫 air.swf 檔案的 launchApplication() 方法，如下列程式碼所示：

```
var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.launchApplication(appID, pubID, arguments);
```

launchApplication() 方法定義於 air.swf 檔案的頂層 (後者是載入到使用者介面 SWF 檔的應用程式網域)。若已安裝指定的應用程式，且應用程式描述器檔案中的 allowBrowserInvocation 設定為允許瀏覽器叫用功能，呼叫此方法將導致 AIR 啟動該應用程式。這個方法具有下列參數：

參數	說明
appID	要啟動之應用程式的應用程式 ID。如需詳細資訊，請參閱 第 193 頁 「id」。
pubID	要啟動之應用程式的發行者 ID。如需詳細資訊，請參閱 第 202 頁 「publisherID」。如果提到的應用程式沒有發行者 ID，請將 pubID 參數設定成空字串 ("")。
arguments	要傳遞至應用程式的引數陣列。應用程式的 NativeApplication 物件會傳送 BrowserInvokeEvent 事件，而該事件的 arguments 屬性即是設定成此陣列。引數中只能辨識英數字母字元。如需傳遞其它值，請考慮使用編碼資料結構。

您必須在使用者事件 (例如按下滑鼠) 的事件處理常式中呼叫 launchApplication() 方法，否則此方法不能運作。

如果瀏覽器並未安裝必要的 Flash Player 版本 (若使用 Windows 和 Mac OS，則安裝第 9 版更新 3 或更新版本；若使用 Linux，則安裝第 10 版)，launchApplication() 方法就會擲出錯誤。

如果應用程式描述器檔案將 allowBrowserInvocation 元素設定為 false，呼叫 launchApplication() 方法就沒有任何效用。

您或許可以先呼叫 air.swf 檔案所提供的 getApplicationVersion() 方法，再決定是否顯示使用者介面告知即將啟動應用程式。如需詳細資訊，請參閱第 219 頁 「從網頁檢查是否已安裝 AIR 應用程式」。

透過瀏覽器叫用功能叫用應用程式時，應用程式的 NativeApplication 物件會傳送 BrowserInvokeEvent 事件。如需詳細資訊，請參閱從瀏覽器叫用 AIR 應用程式 (適用於 ActionScript 開發人員) 或 Invoking an AIR application from the browser (適用於 HTML 開發人員)。

如果您要使用瀏覽器叫用功能，請務必考量安全性方面的隱憂。這些隱憂將描述於從瀏覽器叫用 AIR 應用程式 (適用於 ActionScript 開發人員) 和 Invoking an AIR application from the browser (適用於 HTML 開發人員)。

執行中的應用程式可以使用 LocalConnection 類別，與瀏覽器內的 SWF 內容進行通訊。如需詳細資訊，請參閱與其他 Flash Player 和 AIR 實體進行通訊 (適用於 ActionScript 開發人員) 或 Communicating with other Flash Player and AIR instances (適用於 HTML 開發人員)。

備註：至於 AIR 1.5.3，不建議使用發行者 ID。發行者 ID 將不再自動指派給應用程式。為了回溯相容性，應用程式可以繼續指定發行者 ID。

第 17 章 更新 AIR 應用程式

使用者可以在電腦上按兩下 AIR 檔或 (使用隱藏安裝功能) 從瀏覽器進行 AIR 應用程式的安裝或更新。Adobe® AIR® 安裝程式會管理安裝作業，並在更新現有應用程式時警告使用者

但是，您也可以使用 Updater 類別，讓已安裝的應用程式自行更新至新版本 (已安裝的應用程式可以偵測到已有新版可供下載及安裝)。Updater 類別包含 update() 方法，可以讓您指向使用者電腦上的 AIR 檔，並更新至該版本。您的應用程式必須封裝成 AIR 檔案，才能使用 Updater 類別。封裝成原生執行檔或套件的應用程式應該使用原生平台提供的更新功能。

更新 AIR 檔的應用程式 ID 和發行者 ID 都必須與要更新的應用程式相符。發行者 ID 是從簽署憑證衍生而來。更新和要更新的應用程式都必須以相同的憑證簽署。

對於 AIR 1.5.3 或更新版本，應用程式描述器檔案可包含 <publisherID> 元素。如果有任何應用程式版本是使用 AIR 1.5.2 或更早版本所開發的，您就必須使用此元素。如需詳細資訊，請參閱 第 202 頁 「publisherID」。

從 AIR 1.1 及更新版本開始，您可以移轉應用程式，以使用新的程式碼簽署憑證。移轉應用程式以使用新簽名，包含了同時用新的和原始的憑證來簽署 AIR 更新檔。憑證移轉是單向程序。移轉之後，只有以新憑證 (或以兩種憑證) 簽署的 AIR 檔才會被視為現有應用程式的更新。

管理應用程式更新是一項複雜的工作。AIR 1.5 含有 AdobeAIR 應用程式所適用的全新更新架構。這個架構提供了 API，以協助開發人員賦予 AIR 應用程式良好的更新能力。

您可以使用憑證移轉，從自我簽署憑證變更為商業程式碼簽署憑證，或是從自我簽署或商業憑證變更為其它憑證。如果未移轉憑證，現有使用者就必須先移除目前版本的應用程式，才能安裝新版。如需詳細資訊，請參閱第 165 頁 「變更憑證」。

建議您在應用程式中加入更新機制。如此一來，如果您建立了新版的應用程式，更新機制就會提示使用者安裝新版本。

安裝、更新或移除 AIR 應用程式時，AIR 應用程式安裝程式會建立記錄檔。您可以查閱這些記錄檔，判斷任何安裝問題的發生原因。請參閱安裝記錄。

備註：Adobe AIR 執行階段的新版本可能包含 WebKit 的更新版本。WebKit 的更新版本可能會導致部署的 AIR 應用程式 HTML 內容發生無法預期的變更。這些變更可能會要求您更新應用程式。更新機制可通知使用者有新版的應用程式可用。如需詳細資訊，請參閱關於 HTML 環境 (適用於 ActionScript 開發人員) 或 About the HTML environment (適用於 HTML 開發人員)。

關於更新應用程式

Updater 類別 (在 flash.desktop 套件中) 包含一個方法 update()，可供您將目前執行的應用程式更新為不同的版本。例如，如果使用者桌面具有某個版本的 AIR 檔 ("Sample_App_v2.air")，下列程式碼會更新該應用程式。

ActionScript 範例：

```
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 範例：

```
var updater = new air.Updater();
var airFile = air.File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version = "2.01";
updater.update(airFile, version);
```

在應用程式使用 Updater 類別之前，使用者或應用程式必須先將更新版本的 AIR 檔下載至電腦。如需詳細資訊，請參閱第 224 頁 「將 AIR 檔下載至使用者的電腦」。

呼叫 Updater.update() 方法的結果

當執行階段中的應用程式呼叫 update() 方法時，執行階段會關閉該應用程式，然後嘗試從 AIR 檔安裝新版本。執行階段會檢查 AIR 檔中所指定的應用程式 ID 和發行者 ID 是否與呼叫 update() 方法之應用程式的應用程式 ID 和發行者 ID 相符 (如需有關應用程式 ID 和發行者 ID 的詳細資訊，請參閱 第 173 頁 「AIR 應用程式描述器檔案」)。它也會檢查版本字串是否與傳遞給 update() 方法的 version 字串相符。如果安裝程序順利完成，執行階段就會開啟新版應用程式；否則 (如果安裝程序無法完成)，就會重新開啟現有 (安裝前) 版本應用程式。

若要在 Mac OS 上安裝更新版的應用程式，使用者必須具有適當的系統權限，才能在應用程式目錄內進行安裝。在 Windows 和 Linux 中，使用者必須具有系統管理權限。

如果更新版的應用程式需要更新版的執行階段，則會安裝新的執行階段版本。若要更新執行階段，使用者必須具有電腦的系統管理權限。

測試使用 ADL 的應用程式時，呼叫 update() 方法會產生執行階段例外。

關於版本字串

指定為 update() 方法的 version 參數之字串必須與要安裝 AIR 檔的應用程式描述器檔案中的 version 或 versionNumber 元素中的字串相符。基於安全考量，指定 version 參數為必要的程序。藉由在 AIR 檔中要求應用程式必須確認版本號碼的方式，應用程式將不會誤裝舊的版本 (舊的應用程式版本可能包含目前所安裝應用程式內已修正的安全性弱點)。應用程式也應該以已安裝應用程式中的版本字串，檢查 AIR 檔中的版本字串，以防止降級攻擊。

在 AIR 2.5 之前，版本字串可以是任何格式。例如，可以是 "2.01" 或 "version 2"。但在 AIR 2.5 或更新版本中，版本字串最多可以有三個序列，而每個序列 (以句號分隔) 中最多可有 3 位數。例如，“.0”、“1.0” 和 “67.89.999” 全是有效的版本號碼。您應該在更新應用程式之前，先驗證更新版本字串。

如果 Adobe AIR 應用程式透過網路下載 AIR 檔，那麼最好制定機制，讓網路服務可透過該機制通知下載中的 Adobe AIR 應用程式版本。然後，應用程式就能使用此字串做為 update() 方法的 version 參數。如果 AIR 檔是透過其它方式取得，其中的 AIR 檔版本不明，AIR 應用程式可以檢查 AIR 檔，以判斷版本資訊 (AIR 檔是以 ZIP 壓縮的存檔檔案，而應用程式描述器檔案是存檔檔案中的第二筆記錄)。

如需有關應用程式描述器檔案的詳細資訊，請參閱 第 173 頁 「AIR 應用程式描述器檔案」。

應用程式更新的簽署工作流程

以臨時方式發佈更新會使管理多個應用程式版本的工作變得更加複雜，也會使追蹤憑證的到期日期變得更加困難。憑證可能在您可以發佈更新之前便已過期。

Adobe AIR 執行階段會將已發佈但沒有移轉簽名的應用程式更新視為新的應用程式。使用者必須解除安裝目前的 AIR 應用程式，才能安裝應用程式更新。

若要解決此問題，請以最新的憑證將每個更新的應用程式上傳至不同的部署 URL。請包括提醒您憑證必須在 180 天的寬限期內套用移轉簽名的機制。如需詳細資訊，請參閱第 169 頁 「簽署 AIR 應用程式的更新版本」。

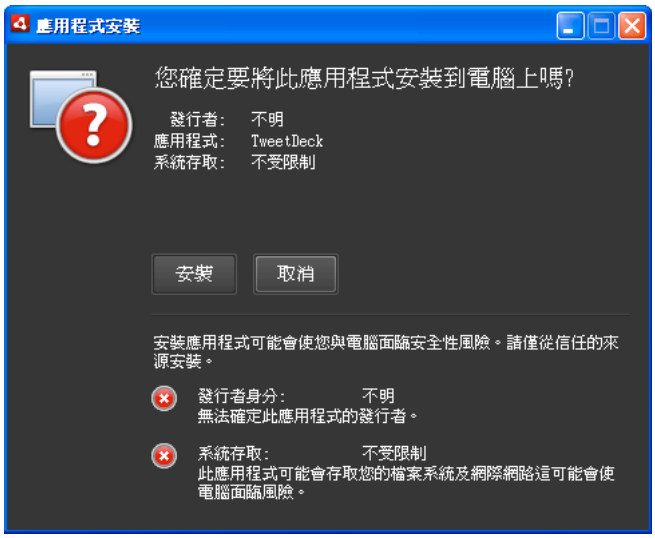
如需有關如何套用簽名的詳細資訊，請參閱 第 142 頁 「ADT 命令」。

執行下列工作以簡化套用移轉簽名的程序：

- 將每個更新的應用程式上傳至不同的部署 URL。
- 將升級描述器 XML 檔案及更新的最新憑證上傳至相同的　URL。
- 使用最新的憑證簽署更新的應用程式。
- 透過用以簽署舊版應用程式的憑證 (位於不同 URL)，將移轉簽名套用至更新的應用程式。

呈現自訂的應用程式更新使用者介面

AIR 包含預設的更新介面：



使用者初次在電腦上安裝一種應用程式版本時，一律會使用此介面，但是，您可以自行定義後續實體要使用的介面。如果您的應用程式定義了自訂的更新介面，請在目前所安裝應用程式的應用程式描述器檔案中指定 customUpdateUI 元素：

```
<customUpdateUI>true</customUpdateUI>
```

安裝應用程式後，使用者透過與已安裝應用程式相符的應用程式 ID 和發行者 ID 來開啟 AIR 檔時，執行階段就會開啟該應用程式，而不會開啟預設 AIR 應用程式安裝程式。如需詳細資訊，請參閱 第 184 頁「customUpdateUI」。

應用程式可以在執行時 (也就是當 NativeApplication.nativeApplication 物件傳送 load 事件時) 決定是否 (使用 Updater 類別) 更新應用程式。如果決定更新，可以向使用者呈現本身的安裝介面 (與其標準執行介面不同)。

將 AIR 檔下載至使用者的電腦

若要使用 Updater 類別，使用者或應用程式必須先以本機方式將 AIR 檔儲存在使用者的電腦上。

備註：AIR 1.5 所具備的更新架構可協助開發人員賦予 AIR 應用程式良好的更新能力。使用這個架構可能比直接使用 Update 類別的 update() 方法簡單得多。如需詳細資訊，請參閱第 227 頁「使用更新架構」。

下列程式碼會從 URL (http://example.com/air/updates/Sample_App_v2.air) 讀取 AIR 檔，並將 AIR 檔儲存在應用程式儲存目錄中。

ActionScript 範例：

```
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 範例：

```
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.");
}
```

如需詳細資訊，請參閱：

- 讀取和寫入檔案的工作流程 (適用於 ActionScript 開發人員)
- Workflow for reading and writing files (適用於 HTML 開發人員)

檢查應用程式是否首次執行

更新應用程式後，您可能會想對使用者顯示 「快速入門」或 「歡迎」訊息。在啟動時，應用程式會檢查是否為首次執行，以便決定是否顯示該訊息。

備註：AIR 1.5 所具備的更新架構可協助開發人員賦予 AIR 應用程式良好的更新能力。這個架構提供了一個簡單的方法，可用來檢查應用程式的某個版本是否為初次執行。如需詳細資訊，請參閱第 227 頁 「使用更新架構」。

執行這項作業的其中一個方式，就是在初始化應用程式時，將檔案儲存在應用程式存放目錄中。每次應用程式啟動時，都應該檢查該檔案是否存在。如果檔案不存在，就表示目前使用者是首次執行此應用程式；如果檔案存在，則表示應用程式已經至少執行過一次。如果檔案存在並包含比目前版本號碼舊的版本號碼，您就知道使用者是首次執行新版本。

下列 Flex 範例示範相關概念：

```
<?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>
```

下列範例示範 JavaScript 中的概念：

```
<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>
```

如果您的應用程式以本機方式儲存資料 (例如，存放在應用程式儲存目錄中)，您可以在首次執行時，檢查先前是否已儲存任何 (舊版) 資料。

使用更新架構

管理應用程式的更新可能是非常冗長而單調的作業。Adobe AIR 應用程式更新架構提供的 API 可讓開發人員在 AIR 應用程式中提供高效的更新功能。AIR 更新架構會為開發人員執行下列工作：

- 根據間隔時間或使用者的要求定期檢查更新
- 從網路來源下載 AIR 檔 (更新項目)

- 在使用者初次執行新安裝版本時提出警示
- 確認使用者是否要檢查更新
- 向使用者顯示新的更新版本相關資訊
- 向使用者顯示下載進度和錯誤資訊

AIR 更新架構為您的應用程式提供了一個簡易的使用者介面樣本。此介面為使用者提供了應用程式更新的基本資訊和組態選項。您的應用程式也可以定義自訂使用者介面，以搭配此更新架構使用。

AIR 更新架構可讓您以簡單的 XML 設定檔儲存 AIR 應用程式更新版本的相關資訊。對大部分應用程式而言，設定這些組態檔並加入若干基本程式碼，便能為使用者提供完善的更新功能。

即使未使用更新架構，Adobe AIR 仍包含了 Updater 類別，可讓 AIR 應用程式使用此類別升級至新版本。此 Updater 類別可以讓應用程式升級成使用者電腦中之 AIR 檔所包含的版本。不過，升級管理所牽涉的作業，可能不僅僅是讓應用程式根據本機儲存的 AIR 檔進行升級即可。

AIR 更新架構檔案

AIR 更新架構包含在 AIR 2 SDK 的 frameworks/libs/air 目錄中。它包含以下檔案：

- applicationupdater.swc：定義更新元件庫的基本功能，適用於 ActionScript。此版本不包含使用者介面。
- applicationupdater.swf：定義更新元件庫的基本功能，適用於 JavaScript。此版本不包含使用者介面。
- applicationupdater_ui.swc：定義更新元件庫的基本功能 (Flex 4 版本)，包括應用程式可用來顯示更新選項的使用者介面。
- applicationupdater_ui.swf：定義更新元件庫的基本功能 (JavaScript 版本)，包括應用程式可用來顯示更新選項的使用者介面。

如需詳細資訊，請參閱以下各節：

- 第 228 頁 「設定您的 Flex 開發環境」
- 第 228 頁 「將架構檔案納入 HTML 類型的 AIR 應用程式」
- 第 229 頁 「基本範例：使用 ApplicationUpdaterUI 版本」

設定您的 Flex 開發環境

AIR 2 SDK 下 frameworks/libs/air 目錄中的 SWC 檔案會定義開發 Flex 和 Flash 時可使用的類別。

若要在以 Flex SDK 進行編譯時使用更新架構，請在針對 amxmlc 編譯器進行的呼叫中納入 ApplicationUpdater.swc 或 ApplicationUpdater_UI.swc 檔案。在下列範例中，編譯器會載入 Flex SDK 目錄內 lib 子目錄中的 ApplicationUpdater.swc 檔案：

```
amxmlc -library-path+=lib/ApplicationUpdater.swc  -- myApp.mxml
```

在下列範例中，編譯器會載入 Flex SDK 目錄內 lib 子目錄中的 ApplicationUpdater_UI.swc 檔案：

```
amxmlc -library-path+=lib/ApplicationUpdater_UI.swc  -- myApp.mxml
```

使用 Flash Builder 進行開發時，請在 「屬性」對話方塊將 SWC 檔案加至 「Flex 建置路徑」設定的 「元件庫路徑」中。

請務必將 SWC 檔案複製至您要在 amxmlc 編譯器 (使用 Flex SDK) 或 Flash Builder 參考的目錄中。

將架構檔案納入 HTML 類型的 AIR 應用程式

更新架構的 frameworks/html 目錄含有下列 SWF 檔：

- applicationupdater.swf— 定義更新元件庫的基本功能，無使用者介面

- applicationupdater_ui.swf— 定義更新元件庫的基本功能，並提供使用者介面供您的應用程式顯示更新選項

AIR 應用程式中的 JavaScript 程式碼可以使用 SWF 檔中所定義的類別。

若要使用更新架構，請將 applicationupdater.swf 或 applicationupdater_ui.swf 檔案放入您的應用程式目錄 (或子目錄) 中。接著，請在即將使用更新架構的 HTML 檔 (以 JavaScript 程式碼撰寫) 中放置一個 script 標籤，以載入檔案：

```
<script src="applicationUpdater.swf" type="application/x-shockwave-flash"/>
```

或者，使用此 script 標記載入 applicationupdater_ui.swf 檔案：

```
<script src="applicationupdater_ui.swf" type="application/x-shockwave-flash"/>
```

這兩個檔案中所定義的 API 將於本文件稍後的部分中描述。

基本範例：使用 ApplicationUpdaterUI 版本

更新架構的 ApplicationUpdaterUI 版本提供了基本介面，可讓您輕鬆應用於應用程式中。以下提供一個基本範例。

首先，建立一個呼叫更新架構的 AIR 應用程式：

1 如果您的應用程式是 HTML 類型的 AIR 應用程式，請載入 applicationupdaterui.swf 檔案：

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

2 在 AIR 應用程式的程式邏輯中，實體化 ApplicationUpdaterUI 物件。

在 ActionScript 中，請使用下列程式碼：

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

在 JavaScript 中，請使用下列程式碼：

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

您可以將此程式碼加入至會在應用程式載入時執行的初始化函數。

3 建立一個名為 updateConfig.xml 的文字檔，然後在檔案中加入下列內容：

```
<?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>
```

編輯 updateConfig.xml 檔案的 URL 元素，使其符合更新描述器檔案在您網站伺服器上的最終位置 (請參閱下一個程序)。

delay 為應用程式檢查更新的間隔天數。

4 將 updateConfig.xml 檔案加入 AIR 應用程式的專案目錄。

5 讓 Updater 物件參照 updateConfig.xml 檔案，並呼叫物件的 initialize() 方法。

在 ActionScript 中，請使用下列程式碼：

```
appUpdater.configurationFile = new File("app:/updateConfig.xml");
appUpdater.initialize();
```

在 JavaScript 中，請使用下列程式碼：

```
appUpdater.configurationFile = new air.File("app:/updateConfig.xml");
appUpdater.initialize();
```

6 建立第二個 AIR 應用程式版本，其版本應與第一個應用程式不同 (版本會在應用程式描述器檔案的 version 元素中指定)。

接著，將 AIR 應用程式更新版本放到您的網站伺服器：

1 將 AIR 檔案的更新版本放到網站伺服器。

2 建立一個名為 updateDescriptor.2.5.xml 的文字檔，然後在檔案中加入下列內容：

```
<?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>
```

編輯 updateDescriptor.xml 檔案的 versionNumber、URL 和 description，使其符合更新 AIR 檔案。使用 AIR 2.5 SDK (及更新的版本) 隨附更新架構的應用程式會使用這個更新描述器格式。

3 建立一個名為 updateDescriptor.1.0.xml 的文字檔，然後在檔案中加入下列內容：

```
<?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>
```

編輯 updateDescriptor.xml 檔案的 version、URL 和 description，使其符合更新 AIR 檔案。使用 AIR 2 SDK (及更早的版本) 隨附更新架構的應用程式會使用這個更新描述器格式。

備註：只有當您要支援在 AIR 2.5 之前建立的應用程式更新時，才需要建立第二個更新描述器檔案。

4 請將 updateDescriptor.2.5.xml 與 updateDescriptor.1.0.xml 檔案加入包含更新 AIR 檔案的相同網站伺服器目錄。

雖然這是一個基本範例，但所提供的更新功能已足以應付許多應用程式的需求。本文件稍後的部分將說明如何讓更新架構最滿足您的需求。

如需其他有關使用更新架構的範例，請參閱位於 Adobe AIR 開發人員中心的下列樣本應用程式：

- Flash 類型應用程式中的更新架構 (http://www.adobe.com/go/learn_air_qs_update_framework_flash_tw)

更新為 AIR 2.5

因為在 AIR 2.5 中已變更指定應用程式版本號碼的規則，所以 AIR 2 更新架構無法剖析 AIR 2.5 應用程式描述器中的版本資訊。此不相容性表示您必須更新應用程式以使用新的更新架構，才能更新應用程式以使用 AIR 2.5 SDK。因此，請將您的應用程式從 2.5 以前的任何 AIR 版本更新為 AIR 2.5 或更新的版本。您需要進行「兩次」更新。第一次更新必須使用 AIR 2 命名空間，並包括 AIR 2.5 更新架構元件庫 (您仍然可以使用 AIR 2.5 SDK 來建立應用程式套件)。第二次更新可以使用 AIR 2.5 命名空間，並包括您應用程式的新功能。

您也可以使用中繼更新，不執行其他動作，只使用 AIR Updater 類別將應用程式直接更新為 AIR 2.5 應用程式。

下列範例說明如何將應用程式從 1.0 版本更新至 2.0 版本。1.0 版本使用舊的 2.0 命名空間。2.0 版本使用 2.5 命名空間，並使用 AIR 2.5 API 實作了一些新功能。

1 以應用程式 1.0 版本為基礎，建立應用程式的中繼版本 1.0.1。

a 在建立應用程式時，請使用 AIR 2.5 應用程式更新程式架構。

備註：若為使用 Flash 技術的 AIR 應用程式，請使用 applicationupdater.swc 或 applicationupdater_ui.swc，若為 HTML 類型的 AIR 應用程式，請使用 applicationupdater.swf 或 applicationupdater_ui.swf。

b 使用舊的命名空間與版本來建立 1.0.1 版本的更新描述器檔案，如下所示：

```
<?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 針對使用 AIR 2.5 API 與 2.5 命名空間的應用程式建立 2.0 版本。

3 建立更新描述器，以便將應用程式從 1.0.1 版本更新至 2.0 版本。

```
<?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>
```

定義更新描述器檔案並將 AIR 檔案加入您的網站伺服器

使用 AIR 更新架構時，您將在更新描述器檔案中定義可用更新的基本資訊，而該檔案會儲存在網站伺服器上。更新描述器檔案是一個簡單的 XML 檔案。應用程式中的更新架構會檢查檔案，以瞭解是否有任何新的版本已經上傳。

AIR 2.5 的更新描述器檔案格式已變更。新格式會使用不同的命名空間。原始命名空間為「http://ns.adobe.com/air/framework/update/description/1.0」。AIR 2.5 命名空間為「http://ns.adobe.com/air/framework/update/description/2.5」。

在 AIR 2.5 之前建立的 AIR 應用程式只能讀取 1.0 版本的更新描述器。使用 AIR 2.5 或更新版本的更新程式架構所建立的 AIR 應用程式只能讀取 2.5 版本的更新描述器。因為這個版本的不相容性，您通常需要建立兩個更新描述器檔案。應用程式 AIR 2.5 版本中的更新邏輯，必須下載使用新格式的更新描述器。舊版的 AIR 應用程式必須繼續使用原始格式。您發行的每個更新都必須修改這兩個檔案 (直到您停止支援在 AIR 2.5 之前建立的版本)。

更新描述器檔案包含下列資料：

- versionNumber—AIR 應用程式的新版本。使用更新描述器中的 versionNumber 元素來更新 AIR 2.5 應用程式。此值必須與新 AIR 應用程式描述器檔案的 versionNumber 元素中所使用的字串相同。如果更新描述器檔案中的版本號碼不符合更新 AIR 檔案的版本號碼，更新架構就會擲出例外。
- version—AIR 應用程式的新版本。使用更新描述器中的 version 元素來更新在 AIR 2.5 之前建立的更新應用程式。此值必須與新 AIR 應用程式描述器檔案的 version 元素中所使用的字串相同。如果更新描述器檔案中的版本不符合更新 AIR 檔案的版本，更新架構就會擲出例外。
- versionLabel— 顯示給使用者看的易懂版本字串。versionLabel 為選擇性項目，但是只能在 2.5 版的更新描述器檔案中指定。如果您在應用程式描述器中使用 versionLabel，請使用它並設定為相同值。
- url— 更新 AIR 檔案的位置。此為包含 AIR 應用程式更新版本的檔案。
- description— 新版本的相關資訊。此項資訊可在更新程序中向使用者顯示。

version 和 url 元素是強制性的元素。description 則是選擇性元素。

以下為 2.5 版更新描述器檔案的樣本：

```
<?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>
```

以下為 1.0 版更新描述器檔案的樣本：

```
<?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>
```

如果您想使用多個語言來定義 description 標籤，請使用定義 lang 特質的多個 text 元素：

```
<?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>
```

將更新描述器檔案隨同更新 AIR 檔案一起放到網站伺服器上。

更新描述器所附的 templates 目錄包含更新描述器檔案的樣本，這些樣本同時提供單一語言和多語言版本。

實體化 Updater 物件

將 AIR 更新架構載入程式碼以後 (請參閱第 228 頁「設定您的 Flex 開發環境」和第 228 頁「將架構檔案納入 HTML 類型的 AIR 應用程式」)，您必須實體化一個 Updater 物件，如下所示。

ActionScript 範例：

```
var appUpdater:ApplicationUpdater = new ApplicationUpdater();
```

JavaScript 範例：

```
var appUpdater = new runtime.air.update.ApplicationUpdater();
```

上述程式碼使用了 ApplicationUpdater 類別 (不提供使用者介面)。如果您想要使用 ApplicationUpdaterUI 類別 (提供使用者介面)，請使用下列範例。

ActionScript 範例：

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

JavaScript 範例：

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

這份文件其餘的程式碼樣本皆假設您已實體化一個名為 appUpdater 的 Updater 物件。

調整更新設定

ApplicationUpdater 和 ApplicationUpdaterUI 都可以透過應用程式所提供的設定檔進行設定，或透過應用程式中的 ActionScript 或 JavaScript 進行設定。

在 XML 設定檔中定義更新設定

更新設定檔為 XML 檔，當中可以含有下列元素：

- updateURL— 字串。代表更新描述器在遠端伺服器上的位置。任何有效的 URLRequest 位置皆可。您必須定義 updateURL 屬性，無論是透過設定檔或透過指令碼定義皆可 (請參閱第 231 頁「定義更新描述器檔案並將 AIR 檔案加入您的網站伺服器」)。您必須先定義這個屬性才能使用更新程式 (亦即，在呼叫 Updater 物件的 initialize() 方法之前；請參閱第 235 頁「初始化更新架構」)。
- delay— 數字。代表檢查更新的時間間隔，以天數表示 (允許使用 0.25 之類的值)。值為 0 (預設) 時表示更新程式不會執行自動定期檢查。

除了 updateURL 和 delay 元素以外，ApplicationUpdaterUI 的設定檔還可以含有下列元素：

- defaultUI：dialog 元素清單。每一個 dialog 元素都有一個 name 特質，該特質會對應到使用者介面中的對話方塊。每一個 dialog 元素都有一個 visible 特質，該特質會定義對話方塊是否要顯示。預設值是 true。name 特質的可能值包括：
  - "checkForUpdate"— 對應至 「檢查更新」、「沒有更新」和 「更新錯誤」對話方塊
  - "downloadUpdate"— 對應至 「下載更新」對話方塊
  - "downloadProgress"— 對應至 「下載進度」和 「下載錯誤」對話方塊
  - "installUpdate"— 對應至 「安裝更新」對話方塊
  - "fileUpdate"— 對應至 「檔案更新」、「檔案沒有更新」和 「檔案錯誤」對話方塊
- "unexpectedError"— 對應至 「未預期的錯誤」對話方塊

  設定為 false 時，對應的對話方塊就不會在更新程序中顯示。

以下提供一個 ApplicationUpdater 架構的設定檔範例：

```
<?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>
```

以下提供一個 ApplicationUpdaterUI 架構的設定檔範例，其中含有 defaultUI 元素的定義：

```
<?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>
```

將 configurationFile 屬性指向該檔案的位置：

ActionScript 範例：

```
appUpdater.configurationFile = new File("app:/cfg/updateConfig.xml");
```

JavaScript 範例：

```
appUpdater.configurationFile = new air.File("app:/cfg/updateConfig.xml");
```

更新架構的 templates 目錄會包含一個設定檔樣本：config-template.xml。

透過 ActionScript 或 JavaScript 程式碼定義更新設定

這些組態參數也可以使用應用程式中的程式碼進行設定，如下所示：

```
appUpdater.updateURL = " http://example.com/updates/update.xml";
appUpdater.delay = 1;
```

Updater 物件的屬性為 updateURL 和 delay。這些屬性所定義的設定，與設定檔內 updateURL 和 delay 元素所定義的設定相同：更新描述器檔案的 URL 以及檢查更新的時間間隔。如果您在程式碼中指定了設定檔 「與」設定，則所有透過程式碼設定的屬性都優先於該屬性在設定檔中所對應的設定。

您必須透過設定檔或透過指令碼定義 updateURL 屬性 (請參閱第 231 頁 「定義更新描述器檔案並將 AIR 檔案加入您的網站伺服器」)，然後才可以使用更新程式 (亦即，在呼叫 Updater 物件的 initialize() 方法之前：請參閱第 235 頁 「初始化更新架構」)。

ApplicationUpdaterUI 架構可定義 Updater 物件的其它屬性，包含：

- isCheckForUpdateVisible— 對應至 「檢查更新」、「沒有更新」和 「更新錯誤」對話方塊
- isDownloadUpdateVisible— 對應至 「下載更新」對話方塊
- isDownloadProgressVisible— 對應至 「下載進度」和 「下載錯誤」對話方塊
- isInstallUpdateVisible— 對應至 「安裝更新」對話方塊
- isFileUpdateVisible— 對應至 「檔案更新」、「檔案沒有更新」和 「檔案錯誤」對話方塊
- isUnexpectedErrorVisible— 對應 「未預期的錯誤」對話方塊

每一個屬性都會對應到 ApplicationUpdaterUI 使用者介面中的一或多個對話方塊。每一個屬性都是一個 Boolean 值，預設值為 true。設定為 false 時，對應的對話方塊就不會在更新程序中顯示。

這些對話方塊屬性會覆寫更新設定檔中的設定。

更新程序

AIR 更新架構會透過下列步驟來完成更新程序：

1 更新程式初始化程序會查看更新檢查作業是否根據所定義的時間間隔執行 (請參閱第 232 頁 「調整更新設定」)。如果已到執行更新檢查作業的時間，更新程序就會接著執行。

2 更新程式下載並解譯更新描述器檔案。

3 更新程式下載更新 AIR 檔案。

4 更新程式安裝應用程式的更新版本。

Updater 物件會在各個步驟完成時傳送事件。在 ApplicationUpdater 版本中，您可以取消各個表示程序步驟成功完成的事件。如果您取消當中的任一事件，下一個程序步驟也會遭到取消。在 ApplicationUpdaterUI 版本中，更新程式會顯示一個對話方塊，讓使用者取消或繼續進行各個程序步驟。

如果您取消了事件，則可以呼叫 Updater 物件的方法，以恢復程序。

在更新程序進行時，Updater 的 ApplicationUpdater 版本會將本身的目前狀態記錄在 currentState 屬性中。這個屬性已設定為字串，可能的值如下：

- "UNINITIALIZED"— 更新程式尚未初始化。
- "INITIALIZING"— 更新程式正在初始化。
- "READY"— 更新程式已經初始化。
- "BEFORE_CHECKING"— 更新程式尚未檢查是否有更新描述器檔案。
- "CHECKING"— 更新程式正在檢查是否有更新描述器檔案。
- "AVAILABLE"— 有可用的更新描述器檔案。
- "DOWNLOADING"— 更新程式正在下載 AIR 檔案。
- "DOWNLOADED"— 更新程式已經下載 AIR 檔案。
- "INSTALLING"— 更新程式正在安裝 AIR 檔案。
- "PENDING_INSTALLING"— 更新程式已經初始化，並有擱置中的更新。

Updater 物件的部分方法只會在更新程式處於特定狀態時執行。

初始化更新架構

在設定了組態屬性之後 (請參閱第 229 頁 「基本範例：使用 ApplicationUpdaterUI 版本」)，請呼叫 initialize() 方法來初始化更新：

```
appUpdater.initialize();
```

此方法會執行下列動作：

- 初始化更新架構，以無訊息的方式同步執行所有擱置中的更新。應用程式必須在啟動時呼叫此方法，因為此方法在接受呼叫時可能會重新啟動應用程式。
- 檢查是否有延緩的更新並加以安裝。
- 更新程序若發生錯誤，便清除應用程式儲存區中的更新檔案和版本資訊。
- 如果更新間隔時間已過期，便啟動更新程序。否則，便重新啟動計時器。

呼叫此方法可能會導致 Updater 物件傳送下列事件：

- UpdateEvent.INITIALIZED— 當初始化完成時即傳送。
- ErrorEvent.ERROR— 當初始化發生錯誤時即傳送。

在傳送 UpdateEvent.INITIALIZED 事件時，更新程序便宣告完成。

當您呼叫 initialize() 方法時，更新程式就會啟動更新程序，並根據時間延遲設定來完成所有步驟。不過，您也可以隨時呼叫 Updater 物件的 checkNow() 方法來啟動更新程序。

```
appUpdater.checkNow();
```

如果更新程序已經在執行，這個方法就不會產生任何結果，否則，便會啟動更新程序。

呼叫 checkNow() 方法會導致 Updater 物件傳送下列事件：

- UpdateEvent.CHECK_FOR_UPDATE 事件會在嘗試下載更新描述器檔案前傳送。

如果您取消了 checkForUpdate 事件，則可以呼叫 Updater 物件的 checkForUpdate() 方法 (請參閱下一節)。如果您沒有取消事件，更新程序就會進行下一個步驟，開始檢查更新描述器檔案。

管理 ApplicationUpdaterUI 版本的更新程序

在 ApplicationUpdaterUI 版本中，使用者可以在使用者介面的對話方塊中按一下 「取消」 按鈕，取消程序。此外，您也可以透過程式設計的方式來取消更新程序，也就是呼叫 ApplicationUpdaterUI 物件的 cancelUpdate() 方法。

您可以設定 ApplicationUpdaterUI 物件的屬性，或定義更新設定檔中的元素，指定更新程式要顯示何種對話方塊以進行確認。如需詳細資訊，請參閱第 232 頁 「調整更新設定」。

管理 ApplicationUpdater 版本的更新程序

您可以呼叫 ApplicationUpdater 物件所傳送事件物件的 preventDefault() 方法，以取消執行更新程序中的步驟 (請參閱第 234 頁 「更新程序」)。取消預設行為時，應用程式會向使用者顯示一則訊息，詢問是否要繼續。

下列各節將說明如何在程序中的步驟遭到取消時，繼續執行更新程序。

下載並解譯更新描述器檔案

ApplicationUpdater 物件會在更新程序開始前傳送 checkForUpdate 事件，也就是在更新程式嘗試下載更新描述器檔案之前。如果您取消 checkForUpdate 事件的預設行為，更新程式就不會下載更新描述檔案。您可以呼叫 checkForUpdate() 方法以繼續更新程序：

```
appUpdater.checkForUpdate();
```

呼叫 checkForUpdate() 方法會使更新程式以非同步方式下載並解譯更新描述器檔案。在呼叫 checkForUpdate() 方法後，Updater 物件可能會傳送下列事件：

- StatusUpdateEvent.UPDATE_STATUS— 更新程式已成功下載並解譯更新描述器檔案。這個事件具有下列屬性：
  - available—Boolean 值。如果有與目前應用程式不同的版本存在，則設定為 true；否則會設定為 false (表示沒有不同的版本)。
  - version— 字串。更新檔案之應用程式描述器檔案的版本
  - details— 陣列。如果沒有當地語系化版本的說明，此陣列便將空字串 ("") 做為所傳回的第一個元素，然後將說明做為所傳回的第二個元素。

    如果一份說明有多種版本 (位於更新描述器檔案)，陣列中就會包含多個子陣列。每一個陣列都具有兩個元素：第一個元素是語言碼 (例如 "en")，第二個元素則是該語言所對應的說明 (字串)。請參閱第 231 頁「定義更新描述器檔案並將 AIR 檔案加入您的網站伺服器」。
- StatusUpdateErrorEvent.UPDATE_ERROR— 發生錯誤，更新程式無法下載或解譯更新描述器檔案。

下載更新 AIR 檔案

ApplicationUpdater 物件會在更新程式成功下載並解譯更新描述器檔案之後傳送 updateStatus 事件。預設的行為是在出現可用的更新檔案時開始下載。如果您取消了預設行為，可以呼叫 downloadUpdate() 方法以繼續更新程序：

```
appUpdater.downloadUpdate();
```

呼叫這個方法會使更新程式以非同步方式下載 AIR 檔案的更新版本。

downloadUpdate() 方法可以傳送下列事件：

- UpdateEvent.DOWNLOAD_START— 與伺服器的連線已經建立。使用 ApplicationUpdaterUI 元件庫時，此事件會顯示一個含有進度列的對話方塊來追蹤下載進度。
- ProgressEvent.PROGRESS— 在檔案下載期間定期傳送。
- DownloadErrorEvent.DOWNLOAD_ERROR— 當連線或下載更新檔案發生錯誤時即傳送。此事件也會在發生無效的 HTTP 狀態時傳送 (例如「404 - 找不到檔案」)。這個事件有一個 errorID 屬性，該屬性為整數，可定義額外錯誤資訊。另一個額外的 subErrorID 屬性可能會包含更多錯誤資訊。
- UpdateEvent.DOWNLOAD_COMPLETE— 更新程式已成功下載並解譯更新描述器檔案。如果您沒有取消這個事件，ApplicationUpdater 版本就會進行下一個步驟，開始安裝更新版本。在 ApplicationUpdaterUI 版本中，會出現一個對話方塊，讓使用者選擇是否要繼續進行。

更新應用程式

當更新檔案的下載作業完成時，ApplicationUpdater 物件就會傳送 downloadComplete 事件。如果您取消了預設行為，則可以呼叫 installUpdate() 方法以繼續更新程序：

```
appUpdater.installUpdate(file);
```

呼叫這個方法會使更新程式安裝 AIR 檔案的更新版本。這個方法包含一個 file 參數，該參數是 File 物件，會參考將做為更新的 AIR 檔案。

呼叫 installUpdate() 方法會使 ApplicationUpdater 物件傳送 beforeInstall 事件：

- UpdateEvent.BEFORE_INSTALL— 在開始安裝更新前傳送。有時候，能夠避免立即安裝本次更新的做法是非常實用的，因為，這麼做的話，使用者就可以在執行更新前先完成目前的工作。呼叫 Event 物件的 preventDefault() 方法可延後安裝作業，直到下次重新啟動為止，在那之前，將不會進行其它更新程序 (包括呼叫 checkNow() 方法後或執行定期檢查後所需執行的更新)。

從任意 AIR 檔案執行安裝

您可以呼叫 installFromAIRFile() 方法，從使用者電腦上的 AIR 檔案安裝更新版本。

```
appUpdater.installFromAIRFile();
```

這個方法會使安裝程式從 AIR 檔案安裝應用程式的更新版本。

installFromAIRFile() 方法可以傳送下列事件：

- StatusFileUpdateEvent.FILE_UPDATE_STATUS— 在 ApplicationUpdater 成功驗證以 installFromAIRFile() 方法傳送的檔案後，傳送此事件。這個事件具有下列屬性：
  - available— 如果有與目前應用程式不同的版本存在，則設定為 true；否則會設定為 false (表示沒有不同的版本)。
  - version— 代表全新可用版本的字串。
  - path— 代表更新檔案的原生路徑。

  如果 StatusFileUpdateEvent 物件的可用屬性設為 true，您便可以取消此事件。取消此事件會一併取消進行更新作業。呼叫 installUpdate() 方法可繼續執行取消的更新作業。
- StatusFileUpdateErrorEvent.FILE_UPDATE_ERROR— 發生錯誤，更新程式無法安裝 AIR 應用程式。

取消更新程序

您可以呼叫 cancelUpdate() 方法以取消更新程序：

```
appUpdater.cancelUpdate();
```

這個方法會取消所有擱置中的下載、刪除所有不完整的下載檔案，並重新啟動定期檢查計時器。

如果 Updater 物件正在初始化，這個方法就不會產生任何結果。

ApplicationUpdaterUI 介面的當地語系化

ApplicationUpdaterUI 類別為更新程序提供了預設的使用者介面，其中包含可讓使用者啟動程序、取消程序以及執行其它相關動作的對話方塊。

更新描述器檔案的 description 元素可讓您以多種語言定義應用程式的說明。您可以使用多個 text 元素定義多個 lang 特質，如下所示：

```
<?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>
```

更新架構使用了最符合使用者當地語系化鍊結的說明。如需詳細資訊，請參閱 「定義更新描述器檔案並將 AIR 檔案加入您的網站伺服器」。

Flex 開發人員可以直接將新語言加入 "ApplicationUpdaterDialogs" 組合包。

JavaScript 開發人員可以呼叫 Updater 物件的 addResources() 方法。這個方法會動態新增語言的資源組合包。資源組合包會定義某語言的當地語系化字串。這些字串將用在各個對話方塊文字欄位中。

JavaScript 開發人員可以使用 ApplicationUpdaterUI 類別的 localeChain 屬性來定義使用者介面所使用的地區鍊。一般來說，只有 JavaScript (HTML) 開發人員會使用這個屬性。Flex 開發人員則可以使用 ResourceManager 來管理地區鍊。

舉例來說，下面的 JavaScript 程式碼即定義羅馬尼亞文和匈牙利文的資源組合包：

```
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;
```

如需詳細資訊，請參閱語言參考中針對 ApplicationUpdaterUI 類別之 addResources() 方法的說明。

第 18 章 檢視原始碼

就像使用者可以檢視在網頁瀏覽器顯示 HTML 頁面的原始碼，使用者也可以檢視 HTML 類型 AIR 應用程式的原始碼。Adobe® AIR® SDK 包含一個 AIRSourceViewer.js JavaScript 檔案，可加至應用程式中以方便使用者檢視原始碼。

載入、設定和開啟來源檢視器

「來源檢視器」程式碼包含在 JavaScript 檔案 AIRSourceViewer.js 中，該檔案位於 AIR SDK 架構目錄內。若要在應用程式中使用 「來源檢視器」，請將 AIRSourceViewer.js 複製至應用程式專案目錄，然後在應用程式的主要 HTML 檔案中使用 script 標籤載入該檔案：

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

AIRSourceViewer.js 檔案定義一個名為 SourceViewer 的類別，您可以從 JavaScript 程式碼中呼叫 air.SourceViewer 來存取該類別。

SourceViewer 類別定義 3 個方法：getDefault()、setup() 和 viewSource()。

方法	說明
getDefault()	靜態方法。可傳回 SourceViewer 實體，供您呼叫其他方法。
setup()	可將組態設定套用至 「來源檢視器」。如需詳細資訊，請參閱第 239 頁 「設定來源檢視器」
viewSource()	可開啟新視窗，供使用者瀏覽和開啟管理應用程式的原始檔案。

備註：使用 「來源檢視器」的程式碼必須位於應用程式安全執行程序中 (在應用程式目錄內的一個檔案中)。

例如，下列 JavaScript 程式碼可實體化 「來源檢視器」物件，並開啟 「來源檢視器」視窗，其中列出所有原始檔案。

```
var viewer = air.SourceViewer.getDefault();
viewer.viewSource();
```

設定來源檢視器

config() 方法可將指定的設定套用至 「來源檢視器」。這個方法有一個參數：configObject。configObject 物件具有屬性，用來定義 「來源檢視器」的組態設定。這些屬性是 default、exclude、initialPosition、modal、typesToRemove 和 typesToAdd。

default

字串指定要顯示在 「來源檢視器」中之初始檔案的相對路徑。

例如，下列 JavaScript 程式碼可開啟 「來源檢視器」視窗，其中包含 index.html 檔案當做初始顯示的檔案。

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.default = "index.html";
viewer.viewSource(configObj);
```

exclude

字串陣列指定要從 「來源檢視器」清單排除的檔案或目錄。路徑是相對於應用程式目錄。不支援萬用字元。

例如，下列 JavaScript 程式碼可開啟 「來源檢視器」視窗，其中列出 AIRSourceViewer.js 檔案以外的所有原始檔案，以及 Images 和 Sounds 子目錄中的檔案：

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.exclude = ["AIRSourceViewer.js", "Images" "Sounds"];
viewer.viewSource(configObj);
```

initialPosition

這個陣列包含兩個數字，指定 「來源檢視器」視窗的初始 x 和 y 座標。

例如，下列 JavaScript 程式碼會在螢幕座標 [40, 60] (X = 40, Y = 60) 的位置開啟 「來源檢視器」視窗。

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.initialPosition = [40, 60];
viewer.viewSource(configObj);
```

modal

這個 Boolean 值指定 「來源檢視器」是否開啟為強制回應 (true) 或非強制回應 (false) 視窗。依預設 「來源檢視器」視窗為強制回應視窗。

例如，下列 JavaScript 程式碼容許使用者同時可以和 「來源檢視器」視窗以及任何應用程式視窗互動的方式開啟 「來源檢視器」視窗：

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.modal = false;
viewer.viewSource(configObj);
```

typesToAdd

這個字串陣列指定要在預設包含的類型之外，另外加入至 「來源檢視器」清單的檔案類型。

依預設 「來源檢視器」清單列出下列檔案類型：

- 文字檔案 — TXT、XML、MXML、HTM、HTML、JS, AS、CSS、INI、BAT、PROPERTIES、CONFIG
- 影像檔案 — JPG、JPEG、PNG、GIF

  如果未指定值，將包含所有預設的類型 (不包括 typesToExclude 屬性中指定的類型)。

  例如，下列 JavaScript 程式碼可開啟包含 VCF 和 VCARD 檔案的 「來源檢視器」視窗：

  ```
  var viewer = air.SourceViewer.getDefault();
  var configObj = {};
  configObj.typesToAdd = ["text.vcf", "text.vcard"];
  viewer.viewSource(configObj);
  ```

  對於每一種要列出的檔案類型，您必須指定 "text" (文字檔案類型) 或 "image" (影像檔案類型)。

typesToExclude

這個字串陣列指定要從 「來源檢視器」排除的檔案類型。

依預設 「來源檢視器」清單列出下列檔案類型：

- 文字檔案 — TXT、XML、MXML、HTM、HTML、JS, AS、CSS、INI、BAT、PROPERTIES、CONFIG
- 影像檔案 — JPG、JPEG、PNG、GIF

  例如，下列 JavaScript 程式碼可開啟 「來源檢視器」視窗，且其中不會列出 GIF 或 XML 檔案。

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToExclude = ["image.gif", "text.xml"];
viewer.viewSource(configObj);
```

對於每一種要列出的檔案類型，您必須指定 "text" (表示文字檔案類型) 或 "image" (表示影像檔案類型)。

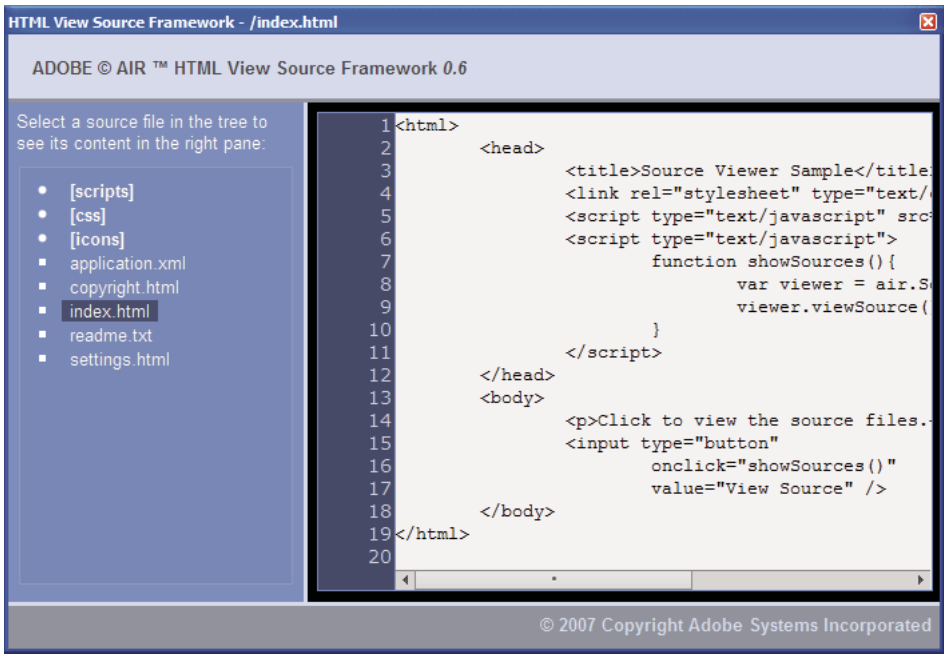
開啟來源檢視器

您應該包含連結、按鈕或選單命令等使用者介面元素，以便當使用者選取時呼叫「來源檢視器」程式碼。例如，下列簡單應用程式會在使用者按一下連結時開啟「來源檢視器」。

```
<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>
```

來源檢視器使用者介面

當應用程式呼叫 SourceViewer 物件的 viewSource() 方法時，AIR 應用程式會開啟 「來源檢視器」視窗。該視窗包含一個原始檔案和目錄的清單 (在左邊)，和一個顯示所選檔案之原始碼的區域 (在右邊)：



目錄列出在括號內。使用者可以按一下括號來展開或收合目錄的清單。

「來源檢視器」可以顯示具有可辨識之副檔名的文字檔案 (例如，HTM、JS、TXT、XML 和其他格式)，或具有可辨識之影像副檔名的影像檔案 (JPG、JPEG、PNG 和 GIF) 的原始碼。如果使用者選取沒有可辨識副檔名的檔案，將會顯示錯誤訊息 (「無法從這個檔案類型擷取文字內容」)。

任何透過 setup() 方法排除的原始檔案將不會列示出來 (請參閱第 239 頁 「載入、設定和開啟來源檢視器」)。

第 19 章 使用 AIR HTML Introspector 除錯

Adobe® AIR® SDK 包含一個名為 AIRIntrospector.js 的 JavaScript 檔案，可加至應用程式中來協助進行 HTML 類型應用程式除錯。

關於 AIR Introspector

Adobe AIR HTML/JavaScript Application Introspector (稱為 AIR HTML Introspector) 提供實用的功能，可協助 HTML 類型應用程式的開發和除錯：

- 它包含一個偵察工具，可以指向應用程式中的使用者介面元素來查看其標記和 DOM 屬性。
- 它還包含一個主控台，可傳送偵察所需的物件參考，而且您可以調整屬性值和執行 JavaScript 程式碼。您還可以將物件序列化至主控台，此時無法編輯資料。您也可以從主控台複製和儲存文字。
- 它包含 DOM 屬性和功能的樹狀檢視。
- 它可以讓您編輯 DOM 元素的特質和文字節點。
- 它可以列出載入至應用程式中的連結、CSS 樣式、影像和 JavaScript 檔案。
- 它可讓您檢視至初始 HTML 來源以及使用者介面的目前標記來源。
- 它可讓您存取應用程式目錄中的檔案。(這個功能僅可用於針對應用程式安全執行程序開啟的 AIR HTML Introspector 主控台。不可用於針對非應用程式安全執行程序內容開啟的主控台。)
- 它包含一個 XMLHttpRequest 物件及其屬性專用的檢視器，包括 responseText 和 responseXML 屬性 (如果可用)。
- 您可以在原始碼和檔案中搜尋相符的文字。

載入 AIR Introspector 程式碼

AIR Introspector 程式碼包含在 JavaScript 檔案 AIRIntrospector.js 中，該檔案位於 AIR SDK 架構目錄內。若要在應用程式中使用 AIR Introspector，請將 AIRIntrospector.js 複製至應用程式專案目錄，然後在應用程式的主要 HTML 檔案中使用 script 標籤載入該檔案。

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

請同時將該檔案加至對應至應用程式中各個不同原生視窗的每一個 HTML 檔案中。

重要事項：請在應用程式開發和除錯階段才加入 AIRIntrospector.js 檔案。請從您要發行的封裝 AIR 應用程式中將其移除。

AIRIntrospector.js 檔案定義一個 Console 類別，您可以從 JavaScript 程式碼中呼叫 air.Introspector.Console 來存取該類型。

備註：使用 AIR Introspector 的程式碼必須位於應用程式安全執行程序中 (在應用程式目錄內的檔案中)。

檢查主控台標籤中的物件

Console 類別定義五個方法：log()、warn()、info()、error() 和 dump()。

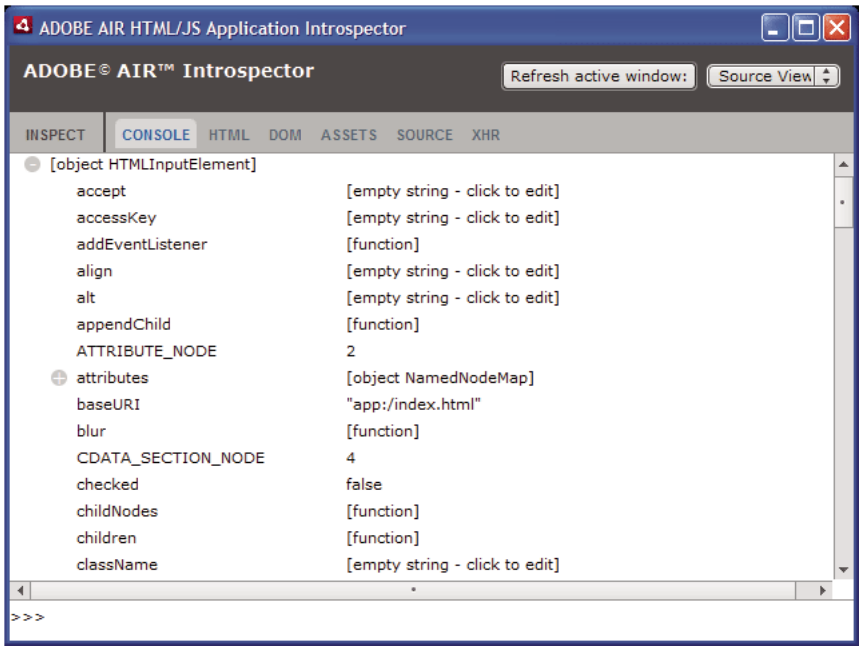
log()、warn()、info() 和 error() 方法都可讓您傳送物件至 「主控台」標籤。log() 是這些方法中最基本的方法。下列程式碼可將 test 變數所表示的簡單物件傳送至 「主控台」標籤：

```
var test = "hello";
air.Introspector.Console.log(test);
```

不過，它更適合用於傳送複雜物件至「主控台」標籤。例如，下列 HTML 頁面包含一個按鈕 (btn1)，可呼叫函數將按鈕本身傳送至「主控台」標籤：

```
<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>
```

當您按一下按鈕，「主控台」標籤會顯示 btn1 物件，您可以展開物件的樹狀檢視來檢查它的屬性：



按一下屬性名稱右邊的項目並修改文字清單，即可編輯物件的屬性。

info()、error() 以及 warn() 方法與 log() 方法相同。不過，當您呼叫這些方法時，「主控台」會在列開頭顯示一個圖示：

方法	圖示
info()	
error()	
warn()	

log()、warn()、info() 和 error() 方法僅會傳送參考至實際物件，因此可用的屬性只限於檢視當時那些屬性。如果您要序列化實際物件，請使用 dump() 方法。這個方法有兩個參數：

參數	說明
dumpObject	要序列化的物件。
levels	物件樹狀結構中要檢查之最大層級數 (除了根層級以外)。預設值為 1 (表示顯示之樹狀結構根層級的後一個層級)。這個參數是選用的參數。

呼叫 dump() 方法先將物件序列化，再傳送至 「主控台」標籤，這樣便無法編輯物件的屬性。例如，以下列程式碼為例：

```
var testObject = new Object();
testObject.foo = "foo";
testObject.bar = 234;
air.Introspector.Console.dump(testObject);
```

當您執行這個程式碼時，主控台會顯示 testObject 物件及其屬性，但您無法在主控台中編輯屬性值。

設定 AIR Introspector

您可以設定全域 AIRIntrospectorConfig 變數的屬性來設定主控台。例如，下列 JavaScript 程式碼可將 AIR Introspector 設定欄每 100 個字便換行：

```
var AIRIntrospectorConfig = new Object();
AIRIntrospectorConfig.wrapColumns = 100;
```

請務必在載入 AIRIntrospector.js 檔案之前先設定 AIRIntrospectorConfig 變數的屬性 (利用 script 標籤)。

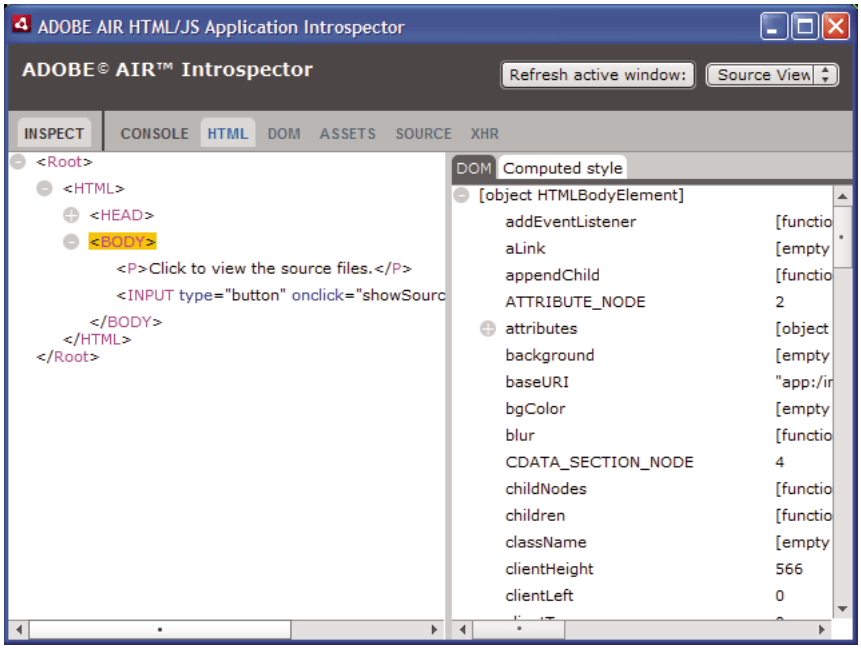
AIRIntrospectorConfig 變數有 8 個屬性：

屬性	預設值	說明
closeIntrospectorOnExit	true	將 Inspector 視窗設定為當應用程式所有其他視窗都關閉時關閉。
debuggerKey	123 (F12 鍵)	顯示和隱藏 AIR Introspector 視窗之鍵盤快速鍵的按鍵代碼。
debugRuntimeObjects	true	將 Introspector 設定為除展開 JavaScript 中定義的物件外，還同時展開執行階段物件。
flashTabLabels	true	將 Console 和 XMLHttpRequest 標籤設定為 flash，以指示它們何時發生變更 (例如，當文字記錄至這些標籤中時)。
introspectorKey	122 (F11 鍵)	開啟檢查畫面之鍵盤快速鍵的按鍵代碼。
showTimestamp	true	將 「主控台」標籤設成在每一行開頭顯示時間戳記。
showSender	true	將 「主控台」標籤設定為在每一列開頭顯示傳送訊息之物件的相關資訊。
wrapColumns	2000	設定原始檔案自動換行的欄數。

AIR Introspector 介面

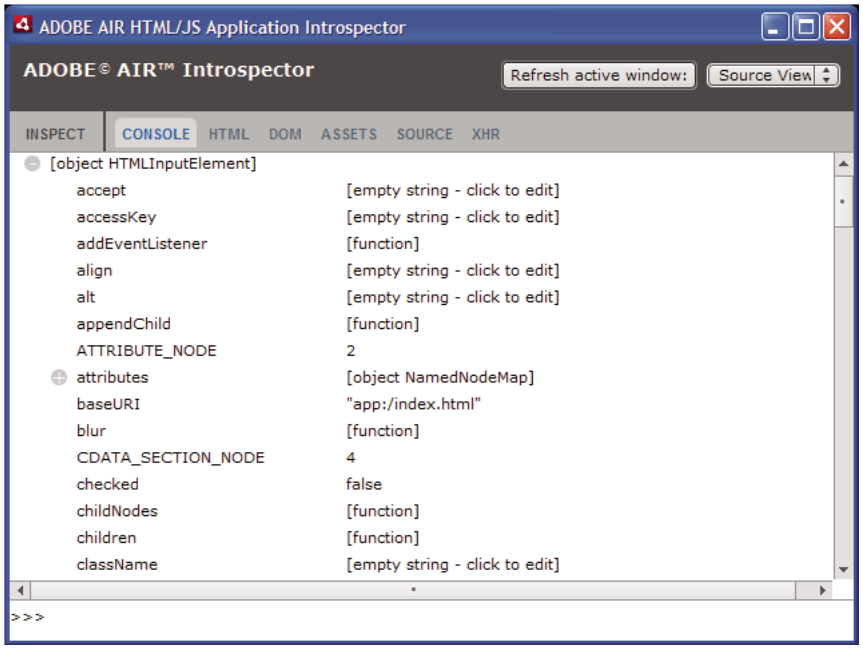
若要在進行應用程式除錯時開啟 AIR Introspector 視窗，請按 F12 鍵或呼叫 Console 類別中的其中一個方法 (請參閱第 243 頁 「檢查主控台標籤中的物件」)。您可以將快速鍵設成 F12 以外的按鍵，請參閱第 245 頁 「設定 AIR Introspector」。

下圖顯示 AIR Introspector 視窗包含的 6 個標籤 — 主控台、HTML、DOM、資源、來源和 XHR：



主控台標籤

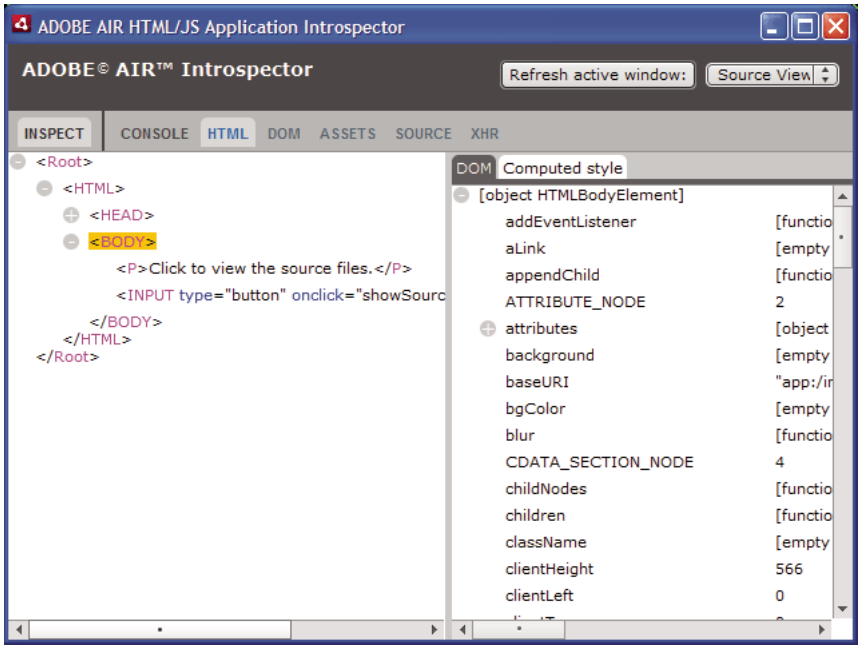
主控台標籤顯示當作參數傳遞至 air.Introspector.Console 類別其中一個方法的內容值。如需詳細資訊，請參閱第 243 頁「檢查主控台標籤中的物件」。



- 若要清除主控台，請以滑鼠右鍵按一下文字，然後選取「清除主控台」。
- 若要將「主控台」標籤中的文字儲存到檔案，請使用滑鼠右鍵按一下「主控台」標籤，然後選取「將主控台儲存到檔案」。
- 若要將「主控台」標籤中的文字儲存至剪貼簿，請使用滑鼠右鍵按一下「主控台」標籤，然後選取「將主控台文字儲存至剪貼簿」。若只要將選取的文字複製至剪貼簿，請以滑鼠右鍵按一下文字，然後選取「複製」。
- 若要將 Console 類別中的文字儲存到檔案，請使用滑鼠右鍵按一下「主控台」標籤，然後選取「將主控台儲存到檔案」。
- 按一下 CTRL+F (在 Windows) 或 Command+F (在 Mac OS) 可以在標籤中搜尋相符的文字 (看不到的樹狀介面節點不會被搜尋)。

HTML 標籤

HTML 標籤可讓您以樹狀結構檢視整個 HTML DOM。按一下元素即可在標籤右邊檢視它的屬性。按一下 + 和 - 圖示可以展開和收合樹狀介面中的節點。



您可以編輯 HTML 標籤中的任何特質或文字元素，編輯的值會反映在應用程式中。

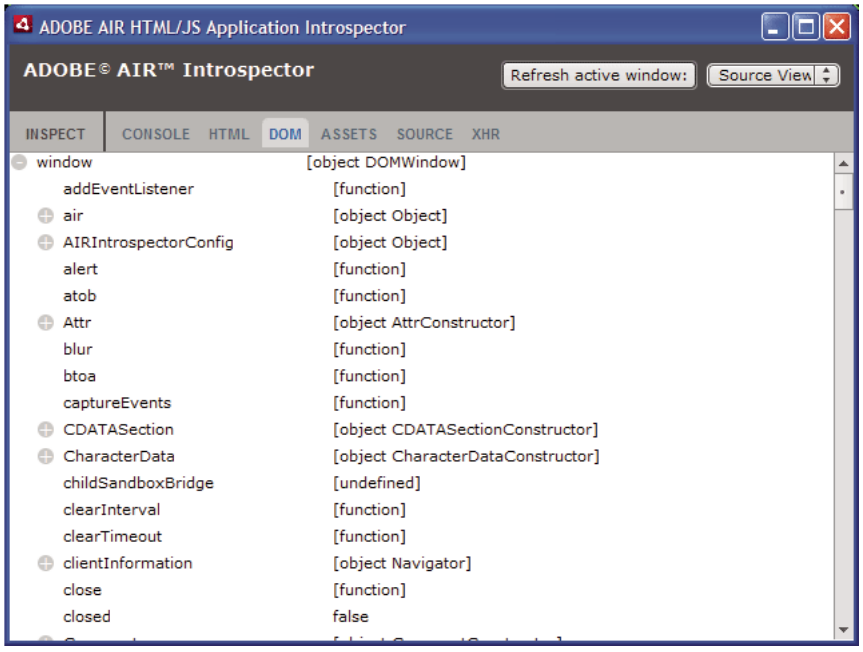
按一下「檢查」按鈕 (AIR Introspector 視窗中標籤清單的最左邊)。您可以按一下主要視窗之 HTML 頁面上的任何元素，和顯示在 HTML 標籤中的關聯 DOM 物件。當主要視窗具有焦點時，您還可以按鍵盤快速鍵來切換開啟或關閉「檢查」按鈕。鍵盤快速鍵預設為 F11。您可以將鍵盤快速鍵設成 F11 鍵以外的按鍵，請參閱第 245 頁「設定 AIR Introspector」。

按一下「重新整理作用中視窗」按鈕 (位於 AIR Introspector 視窗頂端) 可以重新整理顯示在 HTML 標籤中的資料。

按一下 CTRL+F (在 Windows) 或 Command+F (在 Mac OS) 可以搜尋顯示在標籤中的相符文字 (看不到的樹狀介面節點不會被搜尋)。

DOM 標籤

DOM 標籤以樹狀結構顯示視窗物件。您可以編輯任何字串和數值屬性，編輯的值會反映在應用程式中。

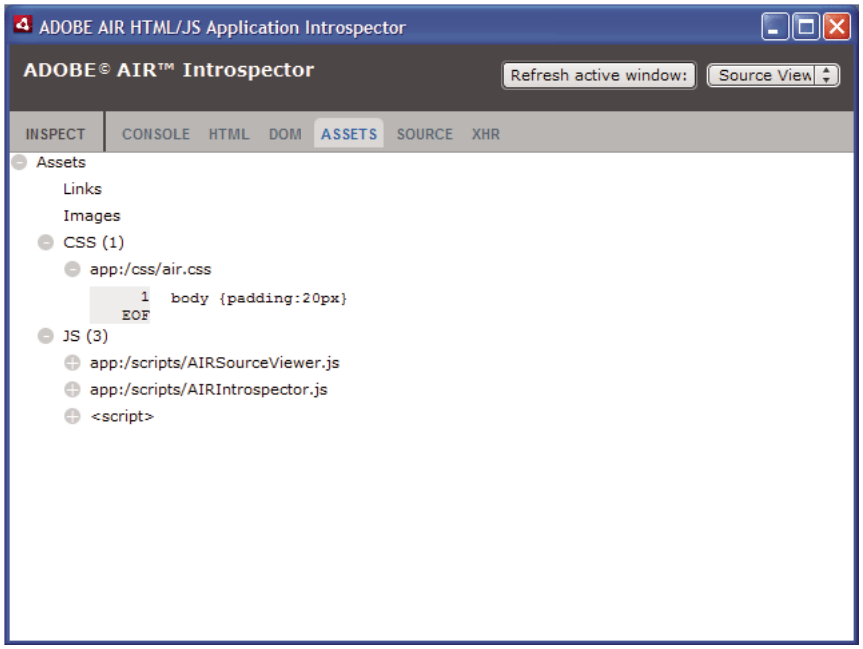


按一下「重新整理作用中視窗」按鈕 (位於 AIR Introspector 視窗頂端)，可以重新整理顯示在 DOM 標籤中的資料。

按一下 CTRL+F (在 Windows) 或 Command+F (在 Mac OS) 可以搜尋顯示在標籤中的相符文字 (看不到的樹狀介面節點不會被搜尋)。

資源標籤

「資源」標籤可讓您檢查載入至原生視窗中的連結、影像、CSS 和 JavaScript 檔案。展開這些節點中的任一個可顯示檔案的內容或使用的實際影像。



按一下「重新整理作用中視窗」按鈕 (位於 AIR Introspector 視窗頂端) 可以重新整理顯示在「資源」標籤中的資料。

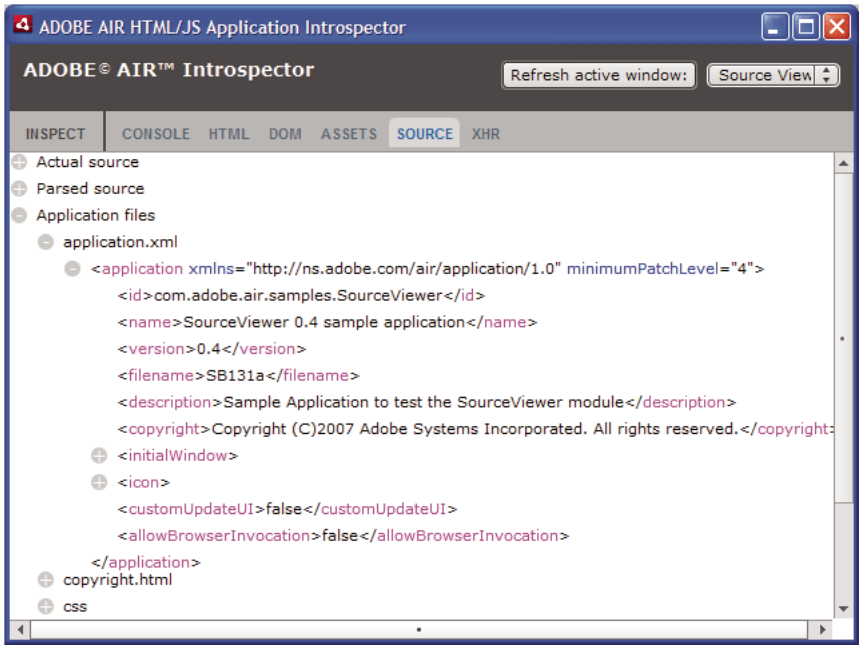
按一下 CTRL+F (在 Windows) 或 Command+F (在 Mac OS) 可以搜尋顯示在標籤中的相符文字 (看不到的樹狀介面節點不會被搜尋)。

來源標籤

「來源」標籤包含三個區段：

- 實際來源 — 顯示應用程式啟動時載入當做根內容之頁面的 HTML 來源。
- 剖析的來源 — 顯示組成目前的應用程式 UI 的標記，可能和實際來源有所不同，因為應用程式使用 Ajax 技術來即時產生標記程式碼。

- 應用程式檔案 — 列出應用程式目錄中的檔案。只有從應用程式安全執行程序中的內容啟動時，這個清單才可以用於 AIR Introspector。在這個區段中，您可以檢視文字檔案的內容或檢視影像。

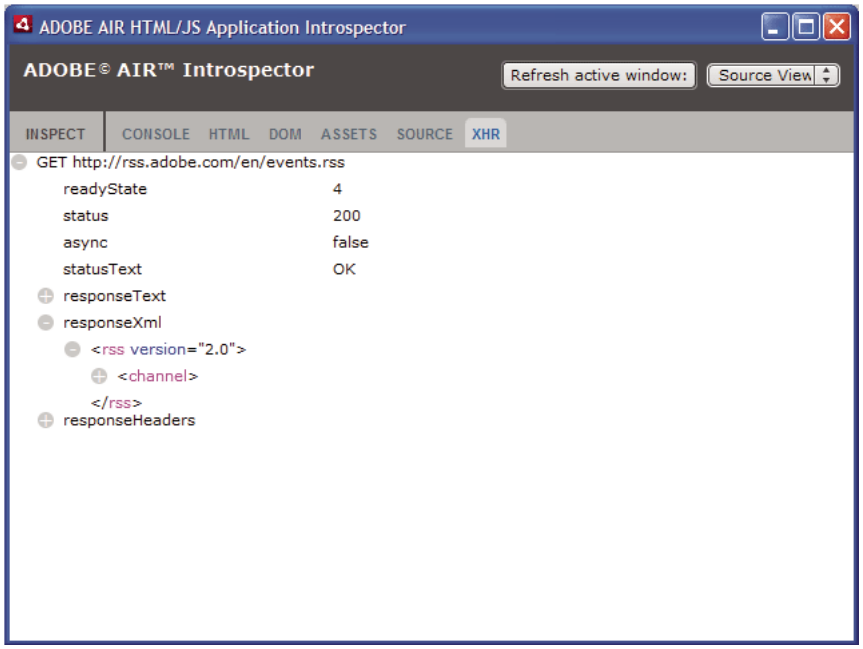


按一下「重新整理作用中視窗」按鈕 (位於 AIR Introspector 視窗頂端) 可以重新整理顯示在「來源」標籤中的資料。

按一下 CTRL+F (在 Windows) 或 Command+F (在 Mac OS) 可以搜尋顯示在標籤中的相符文字 (看不到的樹狀介面節點不會被搜尋)。

XHR 標籤

XHR 標籤可攔截應用程式中的所有 XMLHttpRequest 通訊並記錄該資訊。您因此能夠以樹狀檢視來檢視 XMLHttpRequest 屬性，包括 responseText 和 responseXML (如果可用) 。



按一下 CTRL+F (在 Windows) 或 Command+F (在 Mac OS) 可以搜尋顯示在標籤中的相符文字 (看不到的樹狀介面節點不會被搜尋) 。

對非應用程式安全執行程序中的內容使用 AIR Introspector

您可以將應用程式目錄的內容載入至對應到非應用程式安全執行程序的 iframe 或 frame 中 (請參閱 Adobe AIR 的 HTML 安全性 (適用於 ActionScript 開發人員) 或 HTML security in Adobe AIR (適用於 HTML 開發人員)) 。您可以對這種內容使用 AIR Introspector，但必須遵循下列規則：

- AIRIntrospector.js 檔案必須同時包含在應用程式安全執行程序以及非應用程式安全執行程序 (iframe) 內容中。
- 請勿覆寫 parentSandboxBridge 屬性，因為 AIR Introspector 程式碼要使用這個屬性。請視需要新增屬性。因此，請勿撰寫下列程式碼：

  ```
  parentSandboxBridge = mytrace: function(str) {runtime.trace(str)} ;
  ```

  請使用下列語法：

  ```
  parentSandboxBridge.mytrace = function(str) {runtime.trace(str)};
  ```

- 您無法從非應用程式安全執行程序內容透過按 F12 鍵或呼叫 air.Introspector.Console 類別其中一個方法來開啟 AIR Introspector。您只能透過按一下「開啟 Introspector」按鈕來開啟 Introspector 視窗。這個按鈕預設加在 iframe 或 frame 的右上角。(由於加諸非應用程式安全執行程序內容的安全性限制，新視窗只能由使用者的手勢，例如按一下按鈕來開啟。)

- 您可以針對應用程式安全執行程序和非應用程式安全執行程序個別開啟 AIR Introspector 視窗。您可以透過顯示在 AIR Introspector 視窗中的標題來區分兩者。
- AIR Introspector 是從非應用程式安全執行程序執行時，「來源」標籤不會顯示應用程式檔案
- AIR Introspector 只能查看本身從安全執行程序開啟的程式碼。

第 20 章 當地語系化 AIR 應用程式

Adobe AIR 1.1 及更新的版本

Adobe® AIR® 支援多個語言。

如需當地語系化 ActionScript 3.0 和 Flex Framework 之內容的概觀資訊，請參閱 「ActionScript 3.0 開發人員指南」中的「當地語系化應用程式」。

AIR 中支援的語言

AIR 1.1 版已經引進當地語系化支援，並對 AIR 應用程式提供下列語言：

- 簡體中文
- 繁體中文
- 法文
- 德文
- 義大利文
- 日文
- 韓文
- 巴西葡萄牙文
- 俄文
- 西班牙文

AIR 1.5 版已經加入下列語言：

- 捷克文
- 荷蘭文
- 波蘭文
- 瑞典文
- 土耳其文

更多說明主題

在 Adobe AIR 建立多國語言 Flex 應用程式

建立多國語言 HTML 類型應用程式

當地語系化 AIR 應用程式安裝程式中的應用程式名稱和說明

Adobe AIR 1.1 及更新的版本

您可以在應用程式描述器檔案中，為 name 和 description 元素指定多國語言。例如，在下列程式碼中，會以三種語言 (英文、法文和德文) 指定應用程式的名稱：

```
<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>
```

如 RFC4646 (http://www.ietf.org/rfc/rfc4646.txt) 中所定義，每一個 text 元素的 xml:lang 特質都可以指定一個語言代碼。

name 元素會定義 AIR 應用程式安裝程式所顯示的應用程式名稱。AIR 應用程式安裝程式會針對作業系統設定所定義的使用者介面語言，使用已當地語系化且最適合的值。

同樣地，您也可以在應用程式描述器檔案中，指定 description 元素的多國語言版本。這個元素會定義 AIR 應用程式安裝程式所顯示的說明文字。

這些設定只適用於 AIR 應用程式安裝程式中的可用語言，而不會定義正在執行、所安裝應用程式的可用地區。除了 AIR 應用程式安裝程式可用的語言之外，AIR 應用程式還能提供支援多國語言的使用者介面。

如需詳細資訊，請參閱 第 177 頁 「AIR 應用程式描述器元素」。

更多說明主題

在 Adobe AIR 建立多國語言 Flex 應用程式

建立多國語言 HTML 類型應用程式

使用 AIR HTML 當地語系化組織架構來當地語系化 HTML 內容

Adobe AIR 1.1 及更新的版本

AIR 1.1 SDK 包含 HTML 當地語系化架構。AIRLocalizer.js JavaScript 檔案會定義架構。AIR SDK 的 frameworks 目錄包含 AIRLocalizer.js 檔案。這個檔案中包含 air.Localizer 類別，而這個類別可以協助建立支援多個當地語系化版本的應用程式。

載入 AIR HTML 當地語系化架構程式碼

若要使用當地語系化架構，請將 AIRLocalizer.js 檔案複製到您的專案。接著，再使用下列 script 標籤，將專案包含在應用程式的主要 HTML 檔中：

```
<script src="AIRLocalizer.js" type="text/javascript" charset="utf-8"></script>
```

後續的 JavaScript 可以呼叫 air.Localizer.localizer 物件：

```
<script>
    var localizer = air.Localizer.localizer;
</script>
```

air.Localizer.localizer 是 Singleton 物件，可以定義用來使用和管理已當地語系化之資源的方法和屬性。Localizer 類別包含下列方法：

方法	說明
getFile()	針對指定的地區，取得指定之資源組合包的文字。請參閱第 260 頁 「取得特定地區的資源」。
getLocaleChain()	傳回地區鍊中的語言。請參閱第 259 頁 「定義地區鍊」。
getResourceBundle()	將組合包金鑰和對應值做為物件傳回。請參閱第 260 頁 「取得特定地區的資源」。
getString()	針對某個資源取得已定義的字串。請參閱第 260 頁 「取得特定地區的資源」。

方法	說明
setBundlesDirectory()	設定組合包目錄的位置。請參閱第 259 頁 「自訂 AIR HTML Localizer 設定」。
setLocalAttributePrefix()	設定在 HTML DOM 元素中使用之 localizer 特質的前置詞。請參閱第 259 頁 「自訂 AIR HTML Localizer 設定」
setLocaleChain()	設定地區鍊中的語言順序。請參閱第 259 頁 「定義地區鍊」。
sortLanguagesByPreference()	根據作業系統設定中的地區順序，排序地區鍊中的地區。請參閱第 259 頁 「定義地區鍊」。
update()	使用目前地區鍊中已當地語系化的字串，更新 HTML DOM (或 DOM 元素)。如需有關地區鍊的討論，請參閱第 257 頁 「管理地區鍊」。如需有關 update() 方法的詳細資訊，請參閱第 258 頁 「更新 DOM 元素以使用目前的地區」。

Localizer 類別包含下列靜態屬性：

屬性	說明
localizer	為應用程式傳回 Singleton Localizer 物件的參考。
ultimateFallbackLocale	當應用程式不支援使用者偏好設定時，所使用的地區。請參閱第 259 頁 「定義地區鍊」。

指定支援的語言

在應用程式描述器檔案中使用 <supportedLanguages> 元素，可以識別應用程式支援的語言。這個元素僅限 iOS、Mac 固定執行階段和 Android 應用程式使用，所有其他應用程式類型則會加以忽略。

如果您未指定 <supportedLanguages> 元素，封裝程式預設會根據應用程式類型執行下列動作：

- iOS — AIR 執行階段支援的所有語言會在 iOS App Store 中列為應用程式的支援語言。
- Mac 固定執行階段 — 以固定組合包封裝的應用程式沒有當地語系化資訊。
- Android — 應用程式組合包具有 AIR 執行階段支援之所有語言的資源。

如需詳細資訊，請參閱 第 204 頁 「supportedLanguages」。

定義資源組合包

HTML 當地語系化架構會從 「當地語系化」檔案讀取已當地語系化的字串版本。當地語系化檔案是以索引鍵為基礎之值的集合，並且經過序列化的文字檔案。當地語系化檔案有時稱為 「組合包」。

請為應用程式專案目錄建立名為 locale 的子目錄 (您也可以使用不同的名稱，請參閱第 259 頁 「自訂 AIR HTML Localizer 設定」)。這個目錄將包含當地語系化檔案，因此也稱為 「組合包目錄」。

您可以針對應用程式支援的每個地區，建立組合包目錄的子目錄，並指定每個子目錄的名稱以符合地區碼。例如，將法文目錄命名為 「fr」，以及將英文目錄命名為 「en」。您可以使用底線 (_) 字元，定義具有語言和國家 / 地區碼的地區。例如，您可以將美式英文的目錄命名為 「en_us」 (或者，您也可以使用連字符號而不使用底線，例如 「en-us」。HTML 當地語系化架構可支援這兩個字元)。

您可以將任意數目的資源檔加入至地區子目錄。通常，您必須為每個語言建立當地語系化檔案 (並將檔案置於該語言所代表的目錄中)。HTML 當地語系化架構包含 getFile() 方法，可以讓您讀取檔案的內容 (請參閱第 260 頁 「取得特定地區的資源」)。

副檔名為 .properties 的檔案，就是當地語系化屬性檔案。您可以使用這些檔案，定義地區的索引鍵值配對。屬性檔案會在每一行定義一個字串值。例如，下列程式碼會為一個索引鍵定義 "Hello in English." 字串值，這個索引鍵的名稱為 greeting：

```
greeting=Hello in English.
```

包含下列文字內容的屬性檔案會定義六個索引鍵值配對：

```
title=Sample Application
greeting=Hello in English.
exitMessage=Thank you for using the application.
color1=Red
color2=Green
color3=Blue
```

上述範例為要儲存在 en 目錄中的英文版屬性檔案。

這個屬性檔案的法文版本則會放在 fr 目錄中：

```
title=Application Example
greeting=Bonjour en français.
exitMessage=Merci d'avoir utilisé cette application.
color1=Rouge
color2=Vert
color3=Bleu
```

您可以針對不同的資訊類型，定義多個資源檔。例如，legal.properties 檔案就包含制式法律文字 (例如版權資訊)。您可以在多個應用程式中重複使用該資源。同樣地，您也可以定義不同的檔案，其中每個檔案都會針對使用者介面中不同部分，定義已當地語系化的內容。

這些檔案請使用 UTF-8 編碼，以支援多國語言。

管理地區錬

當應用程式載入 AIRLocalizer.js 檔案時，它會檢查應用程式中定義的地區。這些地區會對應至組合包目錄的子目錄 (請參閱第 256 頁「定義資源組合包」)。這份可用地區清單就稱為「地區錬」。AIRLocalizer.js 檔案會根據作業系統設定所定義的偏好順序，自動排序地區錬 (Capabilities.languages 屬性會依照偏好的順序，列出作業系統使用者介面語言)。

因此，如果應用程式為「en」、「en_US」和「en_UK」地區定義資源，AIR HTML Localizer 架構就會適當地排序地區錬。在回報「en」為主要地區的系統中啟動應用程式時，地區錬就會排序為 ["en", "en_US", "en_UK"]。在這種情況下，應用程式會先搜尋「en」組合包中的資源，然後才輪到「en_US」組合包。

不過，如果系統回報「en-US」為主要地區，則排序時就會使用 ["en_US", "en", en_UK"]。在這種情況下，應用程式會先搜尋「en_US」組合包中的資源，然後才輪到「en」組合包。

根據預設，應用程式會將地區錬中的第一個地區定義為預設的使用地區。您可以在使用者第一次執行應用程式時，詢問使用者並讓使用者選取地區。接著，您可以選擇將所做抉擇儲存在偏好設定檔案中，以便在將來啟動應用程式時，都使用該地區。

您的應用程式可以使用地區錬中任何地區內的資源字串。如果某個特定地區並未定義資源字串，應用程式便會針對地區錬中定義的其它地區使用下一個相符的資源字串。

您可以呼叫 Localizer 物件的 setLocaleChain() 方法，自訂地區錬。請參閱第 259 頁「定義地區錬」。

使用已當地語系化的內容更新 DOM 元素

應用程式中的元素可以參考當地語系化屬性檔案中的索引鍵值。例如，下列範例中的 title 元素會指定 local_innerHTML 特質。當地語系化架構會使用這個特質，搜尋已當地語系化的值。根據預設，這個架構會搜尋以 "local_" 開頭的特質名稱。架構會更新名稱符合 "local_" 之後的文字之特質。在這種情況下，架構會設定 title 元素的 innerHTML 特質。innerHTML 特質會使用預設屬性檔案 (default.properties) 中為 mainWindowTitle 索引鍵所定義的值：

```
<title local_innerHTML="default.mainWindowTitle"/>
```

如果目前的地區未定義任何相符的值，Localizer 架構便會搜尋地區錬的其餘部分。它會使用地區錬中的下一個已定義值所代表的地區。

在下列範例中，p 元素的文字 (innerHTML 特質) 會使用預設屬性檔案中定義之 greeting 索引鍵的值：

```
<p local_innerHTML="default.greeting" />
```

在下列範例中，input 元素的 value 特質 (和已顯示的文字) 會使用預設屬性檔案中定義之 btnBlue 索引鍵的值：

```
<input type="button" local_value="default.btnBlue" />
```

若要更新 HTML DOM 以使用目前地區錬中定義的字串，請呼叫 Localizer 物件的 update() 方法。呼叫 update() 方法可以讓 Localizer 物件剖析 DOM，並套用它尋找當地語系化 ("local_...") 特質的操作：

```
air.Localizer.localizer.update();
```

您可以定義某個特質 (例如「innerHTML」) 及其相對應當地語系化特質 (例如「local_innerHTML」) 的值。在這種情況下，如果當地語系化架構在當地語系化錬結中找到符合特質值的值，就只會覆寫該特質值。例如，下列元素會定義 value 和 local_value 特質：

```
<input type="text" value="Blue" local_value="default.btnBlue"/>
```

您也可以只更新特定 DOM 元素。請參閱下一節第 258 頁 「更新 DOM 元素以使用目前的地區」。

根據預設，對於定義元素之當地語系化設定的特質，AIR HTML Localizer 會使用 "local_" 做為前置詞。例如，local_innerHTML 特質預設會定義用於元素之 innerHTML 值的組合包和資源名稱。此外，local_value 特質預設也會定義當做元素之 value 特質使用的組合包和資源名稱。您可以設定 Localizer 以使用特質前置詞而非 "local_"。請參閱第 259 頁「自訂 AIR HTML Localizer 設定」。

更新 DOM 元素以使用目前的地區

當 Localizer 物件更新 HTML DOM 時，會讓已加上標記的元素根據目前地區錬中定義的字串使用特質值。若要讓 HTML Localizer 更新 HTML DOM，請呼叫 Localizer 物件的 update() 方法：

```
air.Localizer.localizer.update();
```

如果只要更新某個指定的 DOM 元素，請將它當做參數傳遞至 update() 方法。update() 方法只有一個選擇性參數 parentNode。指定 parentNode 參數時，它會定義要當地語系化的 DOM 元素。呼叫 update() 方法並指定 parentNode 參數，即可為指定當地語系化特質的所有子元素設定當地語系化的值。

例如，以下列 div 元素為例：

```
<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>
```

若要更新這個元素，以使用目前地區錬中定義的當地語系化字串，請使用下列 JavaScript 程式碼：

```
var divElement = window.document.getElementById("colorsDiv");
air.Localizer.localizer.update(divElement);
```

如果在地區錬中找不到索引鍵值，當地語系化架構便會將特質值設定為 "local_" 特質的值。例如，在上一個範例中，假設當地語系化架構 (在地區錬的所有 default.properties 檔案中) 找不到 lblColors 索引鍵的值。在這個情況下，這個架構會使用 "default.lblColors" 做為 innerHTML 值。使用這個值，即表示 (告知開發人員) 有遺漏的資源。

update() 方法在地區錬中找不到資源時，便會傳送 resourceNotFound 事件。air.Localizer.RESOURCE_NOT_FOUND 常數會定義 "resourceNotFound" 字串。這個事件有三個屬性：bundleName、resourceName 和 locale。bundleName 屬性是在其中找不到資源之組合包的名稱。resourceName 屬性是在其中找不到資源之組合包的名稱。locale 屬性是在其中找不到資源之地區的名稱。

update() 方法在指定的組合包中找不到資源時，便會傳送 bundleNotFound 事件。air.Localizer.BUNDLE_NOT_FOUND 常數會定義 "bundleNotFound" 字串。這個事件有兩個屬性：bundleName 和 locale。bundleName 屬性是在其中找不到資源之組合包的名稱。locale 屬性是在其中找不到資源之國家 / 地區的名稱。

update() 方法會以非同步方式運作 (因此會以非同步方式傳送 resourceNotFound 和 bundleNotFound 事件)。下列程式碼會設定 resourceNotFound 和 bundleNotFound 事件的事件偵聽程式：

```
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);
}
```

自訂 AIR HTML Localizer 設定

Localizer 物件的 setBundlesDirectory() 方法可以讓您自訂組合包目錄的路徑；而 Localizer 物件的 setLocalAttributePrefix() 方法則能讓您自訂組合包目錄路徑以及 Localizer 所使用的特質值。

預設的組合包目錄會定義為應用程式目錄的地區子目錄，您可以呼叫 Localizer 物件的 setBundlesDirectory() 方法，指定其它目錄。這個方法會採用一個 path 參數，這個參數是所需組合包目錄的路徑，並且以字串表示。path 參數的值可以是下列任何類型：

- String，定義應用程式目錄 (例如 "locales") 的相對路徑
- String，定義使用 app、app-storage 或 file URL 配置 (例如 "app://languages"，請 「不要」使用 http URL 配置) 的有效 URL
- File 物件

如需 URL 與目錄路徑的相關資訊，請參閱：

- File 物件的路徑 (適用於 ActionScript 開發人員)
- Paths of File objects (適用於 HTML 開發人員)

例如，在下列程式碼中，會將組合包目錄設定為應用程式儲存目錄 (而非應用程式目錄) 的 languages 子目錄：

```
air.Localizer.localizer.setBundlesDirectory("languages");
```

請傳遞有效的路徑做為 path 參數，否則，這個方法會擲回 BundlePathNotFoundError 例外。這個錯誤的 name 屬性為 "BundlePathNotFoundError"，而且其 message 屬性會指定無效的路徑。

根據預設，對於定義元素之當地語系化設定的特質，AIR HTML Localizer 會使用 "local_" 做為前置詞。例如，local_innerHTML 特質會定義用於下列 input 元素之 innerHTML 值的組合包和資源名稱：

```
<p local_innerHTML="default.greeting" />
```

Localizer 物件的 setLocalAttributePrefix() 方法可以讓您使用 "local_" 以外的特質前置詞。這個靜態方法會採用一個參數，即您要當做特質前置詞使用的字串。例如，下列程式碼會設定當地語系化架構，以使用 「loc_」做為特質前置詞：

```
air.Localizer.localizer.setLocalAttributePrefix("loc_");
```

您可以自訂當地語系化架構使用的特質前置詞。如果預設值 ("local_") 與程式碼使用的其它特質名稱產生衝突，您可能會想要自訂前置詞。呼叫這個方法時，請務必使用有效的字元做為 HTML 特質的值 (例如，該值不能包含空格字元)。

如需有關在 HTML 元素中使用當地語系化特質的詳細資訊，請參閱第 257 頁 「使用已當地語系化的內容更新 DOM 元素」。

對於不同的應用程式工作階段，組合包目錄和特質前置詞設定也會改變。如果使用自訂的組合包目錄或特質前置詞設定，請務必在每次啟動應用程式時變更這項設定。

定義地區錬

根據預設，當您載入 AIRLocalizer.js 程式碼時，它會使用預設地區錬。這些地區都是組合包目錄中的可用地區，而且作業系統語言設定會定義這個地區錬 (如需詳細資訊，請參閱第 257 頁 「管理地區錬」)。

您可以呼叫 Localizer 物件的靜態 setLocaleChain() 方法修改地區鍊。舉例來說，如果使用者針對特定語言指定偏好設定，您可能會想要呼叫這個方法。setLocaleChain() 方法會採用一個 chain 參數，這個參數是地區的陣列，例如 ["fr_FR","fr","fr_CA"]。陣列中的地區順序會設定架構 (在後續作業中) 搜尋資源的順序。如果在地區鍊中找不到第一個地區的資源，架構便會繼續在其它地區中搜尋資源。如果缺少 chain 引數、該引數不是陣列，或者該引數是空陣列，函數便會失敗，並且擲回 IllegalArgumentsError 例外。

Localizer 物件的靜態 getLocaleChain() 方法會傳回 Array，它會列出目前地區鍊中的所有地區。

下列程式碼會讀取目前的地區鍊，並將兩個法文地區加入至地區鍊的標頭：

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
```

setLocaleChain() 方法會在更新地區鍊時傳送 "change" 事件。air.Localizer.LOCALE_CHANGE 常數會定義 "change" 字串。這個事件具有一個 localeChain 屬性，這個屬性是新地區鍊中的地區碼陣列。下列程式碼會設定這個事件的事件偵聽程式：

```
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);
}
```

靜態 air.Localizer.ultimateFallbackLocale 屬性表示應用程式不支援任何使用者偏好設定時所使用的地區。預設值為 "en"。您可以將這個值設定為其它地區，如下列程式碼所示：

```
air.Localizer.ultimateFallbackLocale = "fr";
```

取得特定地區的資源

Localizer 物件的 getString() 方法會傳回為特定地區中某個資源定義的字串。呼叫這個方法時，不需要指定 locale 值。在這種情況下，這個方法會查看整個地區鍊，並且傳回提供所指定資源名稱之第一個地區中的字串。這個方法具有下列參數：

參數	說明
bundleName	包含資源的組合包。這是不含 .properties 副檔名的屬性檔案名稱 (舉例來說，如果將這個參數設定為 "alerts"，Localizer 程式碼就會在名為 alerts.properties 的當地語系化檔案中查看)。
resourceName	資源名稱。
templateArgs	這是選擇性的。要在取代字串中取代已編號標籤的字串陣列。例如，以對函數的呼叫為例，其中 templateArgs 參數為 ["Raúl", "4"] 且相符的資源字串為 "Hello, {0}。You have {1} new messages."。在這種情況下，函數會傳回 "Hello, Raúl。You have 4 new messages."。若要忽略這項設定，請傳遞 null 值。
locale	這是選擇性的。要使用的地區碼 (例如 "en"、"en_us" 或 "fr")。如果有提供地區但找不到相符的值，這個方法便不會在地區鍊的其它地區中繼續搜尋值。如果未指定地區碼，這個函數會傳回地區鍊中，第一個為指定之資源名稱提供值的地區內的字串。

當地語系化架構可以更新已加上標記的 HTML DOM 特質。不過，您可以透過其它方式，使用已當地語系化的字串。例如，您可以在某些動態產生的 HTML 中使用字串，或者在函數呼叫中，將字串當做參數值使用。例如，下列程式碼會呼叫 alert() 函數時，搭配在 fr_FR 地區之預設屬性檔案的 error114 資源中定義的字串：

```
alert(air.Localizer.localizer.getString("default", "error114", null, "fr_FR"));
```

getString() 方法在指定的組合包中找不到資源時，便會傳送 resourceNotFound 事件。air.Localizer.RESOURCE_NOT_FOUND 常數會定義 "resourceNotFound" 字串。這個事件有三個屬性：bundleName、resourceName 和 locale。bundleName 屬性是在其中找不到資源之組合包的名稱。resourceName 屬性是在其中找不到資源之組合包的名稱。locale 屬性是在其中找不到資源之地區的名稱。

getString() 方法在指定的組合包中找不到資源時，便會傳送 bundleNotFound 事件。air.Localizer.BUNDLE_NOT_FOUND 常數會定義 "bundleNotFound" 字串。這個事件有兩個屬性：bundleName 和 locale。bundleName 屬性是在其中找不到資源之組合包的名稱。locale 屬性是在其中找不到資源之國家 / 地區的名稱。

getString() 方法會以非同步方式運作 (因此會以非同步方式傳送 resourceNotFound 和 bundleNotFound 事件)。下列程式碼會設定 resourceNotFound 和 bundleNotFound 事件的事件偵聽程式：

```
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 物件的 getResourceBundle() 方法會將指定給該地區的組合包傳回。此方法傳回的值是一個物件，該物件的屬性對應到組合包中的金鑰 (如果應用程式找不到指定的組合包，此方法就會傳回 null)。

此方法接受兩個參數 —locale 和 bundleName。

參數	說明
locale	地區 (例如 "fr")。
bundleName	組合包名稱。

例如，下列程式碼會呼叫 document.write() 方法來載入 fr 地區的預設組合包，然後呼叫 document.write() 方法來寫入該組合包的 str1 和 str2 金鑰值。

```
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/>");
```

如果 getResourceBundle() 方法找不到指定的組合包，就會傳送 bundleNotFound 事件。air.Localizer.BUNDLE_NOT_FOUND 常數會定義 "bundleNotFound" 字串。這個事件有兩個屬性：bundleName 和 locale。bundleName 屬性是在其中找不到資源之組合包的名稱。locale 屬性是在其中找不到資源之國家 / 地區的名稱。

Localizer 物件的 getFile() 方法會傳回指定之地區的組合包內容 (以字串表示)。讀取組合包檔案時，會將它當做 UTF-8 檔案來讀取。這個方法包含下列參數：

參數	說明
resourceFileName	資源檔的檔案名稱 (例如 "about.html")。

templateArgs	這是選擇性的。要在取代字串中取代已編號標籤的字串陣列。例如，以對函數的呼叫為例，其中 templateArgs 參數為 ["Raúl", "4"] 且相符的資源檔包含下列兩段訊息：
	
	`<html>`
	`<body>Hello, {0}. You have {1} new messages.</body>`
	`</html>`
	
	在這種情況下，函數會傳回具有這兩段訊息的字串：
	
	`<html>`
	`<body>Hello, Raúl. You have 4 new messages. </body>`
	`</html>`

locale	要使用的地區碼，例如 "en_GB"。如果有提供地區但找不到相符的檔案，這個方法便不會在地區鍊的其它地區中繼續搜尋。如果「未指定」地區碼，函數會傳回地區鍊中，第一個具有符合 resourceFileName 之檔案的地區內的文字。

例如，下列程式碼會使用 fr 地區之 about.html 檔案的內容，呼叫 document.write() 方法。

```
var aboutWin = window.open();
var aboutHtml = localizer.getFile("about.html", null, "fr");
aboutWin.document.close();
aboutWin.document.write(aboutHtml);
```

getFile() 方法在地區鍊中找不到資源時，便會傳送 fileNotFound 事件。air.Localizer.FILE_NOT_FOUND 常數會定義 "resourceNotFound" 字串。getFile() 方法是以非同步方式運作 (因此會以非同步方式傳送 fileNotFound 事件)。這個事件有兩個屬性：fileName 和 locale。fileName 是找不到之檔案的名稱。locale 屬性是在其中找不到資源之地區的名稱。下列程式碼會設定這個事件的事件偵聽程式：

```
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);
}
```

更多說明主題

建立多國語言 HTML 類型應用程式

第 21 章 Path 環境變數

AIR SDK 包含一些可從命令列或終端機啟動的程式。當 Path 環境變數中含有 SDK bin 目錄路徑時，執行這些程式通常比較方便。

在這裡所顯示的資訊討論如何在 Windows、Mac 和 Linux 上設定路徑，希望可以提供您實用的資訊。不過，電腦組態的差異很大，因此此程序並非適用於每個系統。在這些情況下，您可以從您的作業系統文件或網際網路尋找所需的資訊。

使用 Bash 殼層在 Linux 與 Mac OS 上設定 PATH

當您在終端機視窗中輸入命令時，殼層 (讀取您所輸入的內容並嘗試提供適當回應的程式) 必須先找到檔案系統上的命令程式。殼層會在名為 $PATH 的環境變數中尋找儲存之目錄清單中的命令。若要查看目前列在路徑中的項目，請輸入：

```
echo $PATH
```

這將會傳回以冒號分隔的目錄清單，看起來應該如下：

```
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin
```

目標是要將 AIR SDK bin 目錄的路徑加入清單，以便讓殼層可以找到 ADT 與 ADL 工具。假設您已將 AIR SDK 放入 /Users/fred/SDKs/AIR，則下列命令會將所需的目錄加入路徑：

```
export PATH=$PATH:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

備註：如果您的路徑包含空白字元，請以反斜線來逸出，如下所示：

```
/Users/fred\ jones/SDKs/AIR\ 2.5\ SDK/bin
```

若要確認是否可正常運作，您可以再次使用 echo 命令：

```
echo $PATH
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/too
ls
```

到目前為止沒有問題。您現在應該可以輸入下列命令，並收到正確的回應：

```
adt -version
```

如果您已正確修改 $PATH 變數，命令應該會報告 ADT 的版本。

不過，還會有一個問題，在您下次觸發新的終端機視窗時，您會發現路徑中的新項目已不存在。每次啟動新終端機時都必須執行一次設定路徑的命令。

這個問題的常見解決方式是，將命令新增至殼層所使用的其中一個啟動指令碼。在 Mac OS 上，您可以在 ~/username 目錄中建立 .bash_profile 檔案，而每次開啟新終端機視窗時都會執行該檔案。在 Ubuntu 上，會在您啟動 .bashrc 的新終端機視窗時執行啟動指令碼。其他的 Linux 發行套件與殼層程式都有類似的操作方式。

將命令新增至殼層啟動指令碼：

1. 變更為您的主目錄：

   ```
   cd
   ```

2. 請建立殼層組態描述檔 (若有必要)，並使用 “cat >>” 將您輸入的文字重新導向至檔案的結尾。請針對您的作業系統與殼層使用適當的檔案。例如，您可以在 Mac OS 上使用 .bash_profile，在 Ubuntu 上使用 .bashrc。

   ```
   cat >> .bash_profile
   ```

3. 輸入要加入檔案的文字：

   ```
   export PATH=$PATH:/Users/cward/SDKs/android/tools:/Users/cward/SDKs/AIR/bin
   ```

4 在鍵盤上按一下 CTRL-SHIFT-D，即可結束文字重新導向。

5 顯示檔案以確定所有項目都正常：

```
cat .bash_profile
```

6 開啟新終端機視窗以檢查路徑：

```
echo $PATH
```

應該會列出您新增的路徑項目。

如果您稍後在不同的目錄中建立其中一個 SDK 的新版本，請務必更新組態檔案中的路徑命令。否則，殼層將會繼續使用舊版本。

在 Windows 上設定路徑

當您在 Windows 上開啟命令視窗時，該視窗會繼承在系統屬性中定義的全域環境變數。其中一個重要的變數為 path，當您輸入要執行的程式名稱時，path 便是命令程式搜尋的目錄清單。若要在使用命令視窗時查看路徑中目前包括的內容，您可以輸入：

```
set path
```

這將會顯示以分號分隔的目錄清單，看起來如下：

```
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
```

目標是要將 AIR SDK bin 目錄的路徑加入清單，以便讓命令程式可以找到 ADT 與 ADL 工具。假設您已將 AIR SDK 放入 C:\SDKs\AIR，您可以使用下列程序新增適當的路徑項目：

1 請從 「控制台」開啟 「系統內容」對話方塊，或是以滑鼠右鍵按一下 「我的電腦」圖示，然後選擇選單中的 「內容」。

2 在 「進階」索引標籤下，按一下 「環境變數」按鈕。

3 在 「環境變數」對話方塊的 「系統變數」區段中選取 Path 項目

4 按一下 「編輯」。

5 在 「變數值」欄位中捲動至文字的結尾。

6 在目前值的結尾，輸入下列文字：

```
;C:\SDKs\AIR\bin
```

7 按一下所有對話方塊中的 「確定」以儲存路徑。

如果有任何已開啟的命令視窗，請注意這些視窗的環境並未更新。請開啟新的命令視窗，然後輸入下列命令來確定已正確設定路徑：

```
adt -version
```

如果您稍候變更 AIR SDK 的位置，或是新增新的版本，請記得更新 Path 變數。