Skapa ADOBE® AIR®-program

Juridiska meddelanden

Mer information om juridiska meddelanden finns i http://help.adobe.com/sv_SE/legalnotices/index.html.

Innehåll

Kapitel 1: Om Adobe AIR

Adobe® AIR® är en körningsmiljö för flera operativsystem och skärmar där du kan använda dina webbutvecklingskunskaper för att skapa och distribuera RIA-program (Rich Internet Applications) för datorer och mobila enheter. Du kan skapa AIR-program för stationära datorer, tv-enheter och mobiler med ActionScript 3.0 tillsammans med Adobe® Flex och Adobe® Flash® (SWF-baserat). Du kan också skapa AIR-program för stationära datorer med HTML, JavaScript® och Ajax (HTML-baserat).

Mer information om hur du kommer igång med och använder Adobe AIR finns på Adobe AIR Developer Connection (http://www.adobe.com/devnet/air/).

Med AIR kan du arbeta i välbekanta miljöer och använda de verktyg och det tillvägagångssätt som du är van med. Eftersom Flash, Flex, HTML, JavaScript och Ajax stöds kan du arbeta på det sätt som tillgodoser dina behov.

Program kan till exempel utvecklas med hjälp av en eller flera av följande tekniker:

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

Användare använder AIR-program på samma sätt som de använder vanliga datorprogram. Miljön installeras en gång på användarens dator eller enhet, och därefter installeras AIR-programmen och körs precis som andra datorprogram. (I iOS installeras ingen separat AIR-miljö; varje iOS AIR-program är ett självständigt program.)

Runtime-modulen ger en konsekvent plattform och ett konsekvent ramverk för programinstallation på flera operativsystem. Det gör att testning med olika webbläsare inte behövs för att ge enhetliga funktioner och enhetlig användning på alla datorer. Du behöver inte längre utveckla för ett särskilt operativsystem utan kan i stället fokusera på runtime-modulen och ta del av följande fördelar:

- Program som är utvecklade för AIR kan köras på flera olika operativsystem utan att du behöver göra något med dem. Runtime-modulen ger konsekvens och förväntad visning och användning på alla de operativsystem som kan hantera AIR.
- Programmen skapas snabbare eftersom du kan använda befintliga webbtekniker och designmönster. Du kan utöka webbaserade program för stationära datorer utan att du behöver lära dig traditionella utvecklingstekniker eller komplex programkodning.
- Programutvecklingen är enklare än vid användning av lågnivåspråken C eller C++. Du behöver inte hantera specifika, komplexa programmeringsgränssnitt för varje enskilt operativsystem.

När du utvecklar program för AIR kan du använda en mängd olika ramverk och programmeringsgränssnitt:

- Programmeringsgränssnitt för AIR som tillhandahålls av runtime-modulen och AIR-ramverket
- ActionScript-programmeringsgränssnitt som används i SWF-filer och Flex-ramverk (och andra ActionScript-baserade bibliotek och ramverk)
- HTML, CSS och JavaScript
- De flesta Ajax-ramverken
- ANE-tillägg för Adobe AIR ger dig ActionScript-API:er med tillgång till programmeringsbar plattformsspecifik funktionalitet i systemspecifik kod. ANE-tillägg kan även ge tillgång till äldre systemspecifik kod och systemspecifik kod som ger bättre prestanda.

AIR förändrar hur program kan skapas, installeras och användas. Du får större kreativ kontroll och kan utöka Flash-, Flex-, HTML- och Ajax-baserade program för stationära datorer, mobilenheter och tv-enheter.

Information om vad som ingår i varje ny AIR-uppdatering finns i Viktig information om Adobe AIR (http://www.adobe.com/go/learn_air_relnotes_se).

Kapitel 2: Installation av Adobe AIR

I Adobe® AIR®-miljön kan du köra AIR-program. Du kan installera runtime-modulen på följande sätt:

- Genom att installera runtime-modulen separat (utan att installera ett AIR-program)
- Genom att installera ett AIR-program för första gången via en webbsidesinstallation (”badge”) (du uppmanas även att installera körningsmiljön)
- Genom att skapa ett eget installationsprogram som både installerar programmet och miljön. Du måste ha tillstånd från Adobe för att distribuera AIR-miljön på detta sätt. Du begär tillstånd på sidan Adobe runtime licensing. Observera att Adobe inte tillhandahåller verktyg för att skapa sådana installationsprogram. Det finns emellertid många installationsprogram från tredjepartsleverantörer.
- Genom att installera ett AIR-program som paketerar AIR som en låst miljö. En låst miljö används endast i det paketerade programmet. Det används inte för att köra andra AIR-program. Att paketera miljön är ett alternativ i Mac och Windows. I iOS innehåller alla program en paketerad miljö. Från och med AIR 3 innehåller alla Android-program som standard en paketerad miljö (men du kan välja att använda ett separat miljö).
- Genom att ställa in en AIR-utvecklingsmiljö, t.ex. AIR SDK, Adobe® Flex® Builder™ eller Adobe Flex® SDK (som omfattar utvecklingsverktyg för AIR-kommandoraden). Miljön som ingår i SDK används bara vid felsökning av program, inte för att köra installerade AIR-program.

Systemkraven för installation av AIR och körning av AIR-program anges här: Adobe AIR: Systemkrav (http://www.adobe.com/se/products/air/systemreqs/).

Både körtidsmodulens och AIR-programmets installationsprogram skapar loggfiler när AIR-program eller själva AIR-körtidsmodulen installeras, uppdateras eller tas bort. Du kan studera loggarna för att fastställa orsaken till eventuella installationsproblem. Läs mer i Installation logs.

Installera Adobe AIR

Användaren måste ha administratörsbehörighet för datorn för att kunna installera eller uppdatera miljön.

Installera miljön på en Windows-dator

1. Hämta installationsfilen för miljön från http://get.adobe.com/air.
2. Dubbelklicka på installationsfilen.
3. Slutför installationen genom att följa de anvisningar som visas i installationsfönstret.

Installera runtime-modulen på en Mac-dator

1. Hämta installationsfilen för miljön från http://get.adobe.com/air.
2. Dubbelklicka på installationsfilen.
3. Slutför installationen genom att följa de anvisningar som visas i installationsfönstret.
4. Om installationsprogrammet visar ett verifieringsfönster anger du ditt användarnamn och lösenord för Mac OS.

Installera runtime-modulen på en Linux-dator

Obs! *För närvarande stöds inte AIR 2.7 och senare på Linux. AIR-program som installeras på Linux ska fortsättningsvis använda AIR 2.6 SDK.*

Med det binära installationsprogrammet:

1 Leta reda på den binära installationsfilen på http://kb2.adobe.com/se/cps/853/cpsid_85304.html och hämta den.

2 Ange filbehörigheterna så att installationsprogrammet kan köras. Från kommandoraden kan du ange filbehörighet med:

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

I en del versioner av Linux kan du ange filbehörigheterna i dialogrutan Properties som du öppnar via en snabbmeny.

3 Kör installationsprogrammet från kommandoraden eller genom att dubbelklicka på installationsfilen för runtime-modulen.

4 Slutför installationen genom att följa de anvisningar som visas i installationsfönstret.

Adobe AIR installeras som ett internt paket. Med andra ord som RPM på en RPM-baserad distribution och som DEB på en Debian-distribution. AIR har i nuläget inte stöd för några andra paketformat.

Med paketinstallerare:

1 Leta reda på AIR-paketeringsfilen på http://kb2.adobe.com/se/cps/853/cpsid_85304.html. Beroende på vilket paketformat du har ska du hämta RPM- eller Debian-paketet.

2 Vid behov dubbelklickar du på AIR-paketfilen för att installera paketet.

Du kan också installera via kommandotolken:

a På ett Debian-baserat system:

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

b På ett RPM-baserat system:

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

Eller, om du uppdaterar en befintlig version (AIR 1.5.3 eller senare):

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

Du måste ha administratörsbehörighet för datorn för att kunna installera AIR 2 och AIR-program.

Adobe AIR installeras på följande plats: /opt/Adobe AIR/Versions/1.0

AIR registrerar mime-typen "application/vnd.adobe.air-application-installer-package+zip", vilket innebär att AIR-filer får den här mime-typen och därför även registreras med AIR-miljön.

Installera miljön på en Android-enhet

Du kan installera den senaste versionen av AIR-miljön från Android Market.

Du kan installera utvecklingsversioner av AIR-miljön från en länk på en webbsida eller med hjälp av ADT-kommandot `-installRuntime`. Du kan bara installera en version av AIR-miljön i taget. Du kan inte ha både en officiell version och en utvecklingsversion installerade samtidigt.

Läs mer i ”ADT-kommandot installRuntime” på sidan 174.

Installera miljön på en iOS-enhet

Den nödvändiga koden för AIR-miljön paketeras med alla program som skapas för iPhone-, iTouch- och iPad-enheter. Du behöver inte installera någon separat miljökomponent.

Fler hjälpavsnitt

”AIR for iOS” på sidan 68

Ta bort Adobe AIR

När du har installerat runtime-modulen kan du ta bort dem med följande procedurer.

Ta bort runtime-modulen på en dator med Windows

1. Gå till Start-menyn, välj Inställningar > Kontrollpanelen.
2. Öppna Program, Program och funktioner eller Lägg till eller ta bort program på Kontrollpanelen (beroende på vilken version av Windows du har).
3. Välj "Adobe AIR" för att ta bort runtime-modulen.
4. Klicka på knappen Ändra/ta bort.

Ta bort runtime-modulen på en dator med Mac

- Dubbelklicka på "Avinstallationsprogram för Adobe AIR" som finns i mappen /Program/Verktygsprogram.

Ta bort runtime-modulen på en dator med Linux

Gör något av följande:

- Välj kommandot "Avinstallera Adobe AIR" på programmenyn.
- Kör binärfilen för AIR-installationen med alternativet `-uninstall`.
- Ta bort AIR-paketen (`adobeair` och `adobecerts`) med pakethanteraren.

Ta bort miljön från en Android-enhet

1. Öppna programmet med inställningar på enheten.
2. Knacka på Adobe AIR under Applications > Manage Applications.
3. Knacka på knappen Uninstall.

Du kan också använda ADT-kommandot `-uninstallRuntime`. Läs mer i "ADT-kommandot uninstallRuntime" på sidan 175.

Ta bort en paketerad miljö

Om du vill ta bort en låst paketerad miljö måste du ta bort programmet som den är installerat med. Observera att låsta miljöer endast används för att köra installationsprogrammet.

Installera och köra AIR-exempelprogram

Användaren måste ha administratörsbehörighet för datorn för att kunna installera eller uppdatera ett AIR-program.

Det finns exempelprogram som visar funktionerna i AIR. Följ anvisningarna nedan om du vill installera dem:

1. Hämta och kör AIR-exempelprogrammen. Både de kompilerade programmen och källkoden är tillgängliga.
2. Om du vill hämta och köra ett exempelprogram klickar du på knappen Installera nu för exempelprogrammet. Du uppmanas att installera och köra programmet.
3. Om du väljer att hämta exempelprogrammen och köra dem senare väljer du hämtningslänkarna. Du kan köra AIR-programmen när som helst.
   - I Windows: Dubbelklicka på programikonen på skrivbordet eller välj programmet från Start-menyn.

- I Mac OS: Dubbelklicka på programikonen som är installerad i programmappen i användarkatalogen (t.ex. i Macintosh HD/Users/JoeUser/Applications/).

Obs! *Läs igenom AIR Release Notes och se om det finns några uppdateringar för de här anvisningarna. Du hittar dem här: http://www.adobe.com/go/learn_air_relnotes_se.*

Adobe AIR-uppdateringar

Adobe uppdaterar Adobe AIR regelbundet med nya funktioner eller åtgärder för att lösa smärre problem. Funktionen för automatiska meddelanden och uppdateringar används för att meddela användare när en Adobe AIR-uppdatering är tillgänglig.

Adobe AIR-uppdateringarna är en garanti för att Adobe AIR fungerar som det är tänkt, och de innehåller ofta viktiga säkerhetsändringar. Adobe rekommenderar att du uppdaterar till den senaste versionen av Adobe AIR när en ny version finns tillgänglig. Detta är speciellt viktigt när det kommer en uppdatering som avser säkerhet.

Standard är att när AIR-programmet startas så sker automatiskt en kontroll om det finns några tillgängliga uppdateringar. Den här kontrollen görs om det gått mer än två veckor sedan den senaste uppdateringskontrollen. Om det finns en uppdatering kommer den att laddas ner i bakgrunden.

Du kan inaktivera den automatiska uppdateringen genom att använda AIR SettingsManager-programmet. Du kan hämta AIR SettingsManager-programmet på http://airdownload.adobe.com/air/applications/SettingsManager/SettingsManager.air.

Den normala installationsprocessen för Adobe AIR innebär att en anslutning till http://airinstall.adobe.com görs för att skicka basinformation om installationsmijön, till exempel om operativsystemet och språket. Denna information skickas bara en gång per installation och den används av Adobe för att bekräfta att installationen lyckades. Inga personidentifierande uppgifter samlas in eller överförs.

Uppdatera låsta miljöer

Om du distribuerar programmet med ett låst miljöpaket, kommer det låsta miljöpaketet inte att uppdateras automatiskt. För att skydda dina användare måste du kontrollera de uppdateringar som publiceras av Adobe och uppdatera ditt program med nya miljöversioner när en relevant säkerhetförändring publiceras.

Kapitel 3: Arbeta med programmeringsgränssnitten (API:erna) i AIR

I Adobe® AIR® finns funktioner som inte är tillgängliga för SWF-innehåll som körs i Adobe® Flash® Player.

ActionScript 3.0-utvecklare
Programmeringsgränssnitten i Adobe AIR beskrivs i följande två handböcker:

- Utvecklarhandbok för ActionScript 3.0
- Referenshandbok för ActionScript 3.0 i Adobe Flash-plattformen

HTML-utvecklare
Om du skapar HTML-baserade AIR-program beskrivs de API:er som är tillgängliga i JavaScript via filen AIRAliases.js (se Accessing AIR API classes from JavaScript) i följande två handböcker:

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

AIR-specifika ActionScript 3.0-klasser

Följande tabell visar de körningsklasser som är specifika för Adobe AIR. De är inte tillgängliga för SWF-innehåll som körs i Adobe® Flash® Player i webbläsaren.

HTML-utvecklare
De klasser som är tillgängliga i JavaScript via filen AIRAliases.js visas i Adobe AIR API Reference for HTML Developers.

Klass	ActionScript 3.0-paket	Tillagt i AIR-version
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

Klass	ActionScript 3.0-paket	Tillagt i AIR-version
CameraRollBrowseOptions	flash.media	3.0
CameraUI	flash.media	2.5
CertificateStatus	flash.security	2.0
CompressionAlgorithm	flash.utils	1.0
DatagramSocket	flash.net	2.0
DatagramSocketDataEvent	flash.events	2.0
DNSResolver	flash.net.dns	2.0
DNSResolverEvent	flash.events	2.0
DockIcon	flash.desktop	1.0
DownloadErrorEvent	air.update.events	1.5
DRMAuthenticateEvent	flash.events	1.0
DRMDeviceGroup	flash.net.drm	3.0
DRMDeviceGroupErrorEvent	flash.net.drm	3.0
DRMDeviceGroupEvent	flash.net.drm	3.0
DRMManagerError	flash.errors	1.5
EncryptedLocalStore	flash.data	1.0
ExtensionContext	flash.external	2.5
File	flash.filesystem	1.0
FileListEvent	flash.events	1.0
FileMode	flash.filesystem	1.0
FileStream	flash.filesystem	1.0
FocusDirection	flash.display	1.0
GameInput	flash.ui	3.0
GameInputControl	flash.ui	3.0
GameInputControlType	flash.ui	3.6 och tidigare; inte från 3.7
GameInputDevice	flash.ui	3.0
GameInputEvent	flash.ui	3.0
GameInputFinger	flash.ui	3.6 och tidigare; inte från 3.7
GameInputHand	flash.ui	3.6 och tidigare; inte från 3.7
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

Klass	ActionScript 3.0-paket	Tillagt i AIR-version
HTMLPDFCapability	flash.html	1.0
HTMLSWFCapabiltiy	flash.html	2.0
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

Klass	ActionScript 3.0-paket	Tillagt i AIR-version
NativeWindowResize	flash.display	1.0
NativeWindowSystemChrome	flash.display	1.0
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

Klass	ActionScript 3.0-paket	Tillagt i AIR-version
SQLMode	flash.data	1.0
SQLResult	flash.data	1.0
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

Flash Player-klasser med AIR-specifik funktionalitet

Följande klasser är tillgängliga för SWF-innehåll som körs i webbläsaren, men AIR omfattar fler egenskaper eller metoder:

Paket	Klass	Egenskap, metod eller händelse	Tillagt i AIR-version
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-händelse	2.0
		orientationChanging-händelse	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

Paket	Klass	Egenskap, metod eller händelse	Tillagt i AIR-version
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

Paket	Klass	Egenskap, metod eller händelse	Tillagt i AIR-version
flash.net	FileReference	`extension`	1.0
		`httpResponseStatus`-händelse	1.0
		`uploadUnencoded()`	1.0
	NetStream	`drmAuthenticate`-händelse	1.0
		`onDRMContentData`-händelse	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	`händelsen httpResponseStatus`	1.0

Paket	Klass	Egenskap, metod eller händelse	Tillagt i AIR-version
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
flash.system	Capabilities	`språk`	1.1
	LoaderContext	`allowLoadBytesCodeExecution`	1.0
	Security	`APPLICATION`	1.0
flash.ui	KeyLocation	D_PAD	2.5

De flesta av dessa nya egenskaper och metoder är bara tillgängliga för innehåll i AIR-säkerhetssandlådan application. De nya medlemmarna i URLRequest-klasserna är emellertid också tillgängliga för innehåll som körs i andra sandlådor.

Metoderna `ByteArray.compress()` och `ByteArray.uncompress()` omfattar båda en ny `algorithm`-parameter som gör att du kan välja mellan dekomprimering och zlib-komprimering. Den här parametern är endast tillgänglig för innehåll som körs i AIR.

AIR-specifika Flex-komponenter

Följande Adobe® Flex™ MX-komponenter är tillgängliga vid utveckling av innehåll för Adobe AIR:

- FileEvent
- FileSystemComboBox
- FileSystemDataGrid
- FileSystemEnumerationMode

- FileSystemHistoryButton
- FileSystemList
- FileSystemSizeDisplayMode
- FileSystemTree
- FlexNativeMenu
- HTML
- Window
- WindowedApplication
- WindowedSystemManager

Dessutom innehåller Flex 4 följande AIR-komponenter med spark:

- Window
- WindowedApplication

Mer information om AIR Flex-komponenter finns i Using the Flex AIR components.

Kapitel 4: Adobe Flash plattformsverktyg för AIR-utveckling

Du kan utveckla AIR-program med följande utvecklingsverktyg för Adobe Flash-plattformen.

För utvecklare som använder ActionScript 3.0 (Flash och Flex):

- Adobe Flash Professional (se Publicera för Adobe AIR)
- SDK:erna för Adobe Flex 3.x och 4.x (se ”Ställa in Flex SDK” på sidan 19 och ”AIR Developer Tool (ADT)” på sidan 162)
- Adobe Flash Builder (se Developing AIR Applications with Flash Builder)

För HTML- och Ajax-utvecklare:

- Adobe AIR SDK (se ”Installera AIR SDK” på sidan 17 och ”AIR Developer Tool (ADT)” på sidan 162)
- Adobe Dreamweaver CS3, CS4, CS5 (se AIR-tillägget för Dreamweaver)

Installera AIR SDK

Adobe AIR SDK innehåller följande kommandoradsverktyg som du kan använda för att starta och paketera program:

AIR Debug Launcher (ADL) Gör att du kan köra AIR-program utan att först installera dem. Läs mer i ”AIR Debug Launcher (ADL)” på sidan 156.

AIR Development Tool (ADT) Paketerar AIR-program som installationspaket för distribution. Läs mer i ”AIR Developer Tool (ADT)” på sidan 162.

För AIR kommandoradsverktygen måste Java vara installerat på datorn. Du kan använda en JVM (Java Virtual Machine) från antingen JRE eller JDK (version 1.5 eller senare). Java JRE och Java JDK finns på http://java.sun.com/.

Det krävs minst 2 GB minne för att köra ADT-verktyget.

Obs! *Java krävs inte för att slutanvändare ska kunna köra AIR-program.*

En snabb genomgång av hur du skapar ett AIR-program med AIR SDK finns i ”Skapa ditt första HTML-baserade AIR-program med AIR SDK” på sidan 32.

Hämta och installera AIR SDK

Du kan hämta och installera AIR SDK med hjälp av instruktionerna som följer:

Installera AIR SDK i Windows

- Hämta AIR SDK installationsfilen.
- AIR SDK distribueras som ett vanligt filarkiv. Extrahera innehållet i SDK till en mapp på datorn (t.ex. C:\Programfiler\Adobe\AIRSDK eller C:\AIRSDK) när du ska installera AIR.
- Verktygen ADL och ADT finns i mappen bin i AIR SDK. Lägg till sökvägen till mappen i systemvariabeln PATH.

Installera AIR SDK i Mac OS X

- Hämta AIR SDK installationsfilen.
- AIR SDK distribueras som ett vanligt filarkiv. Extrahera innehållet i SDK till en mapp på datorn (t.ex. /Users/<användarnamn>/Applications/AIRSDK) när du ska installera AIR.
- Verktygen ADL och ADT finns i mappen bin i AIR SDK. Lägg till sökvägen till mappen i systemvariabeln PATH.

Installera AIR SDK i Linux

- SDK är tillgängligt i formatet tbz2.
- Skapa en mapp där du vill extrahera SDK och använd sedan följande kommando: tar -jxvf <sökväg till AIR-SDK.tbz2> när du ska installera SDK.

Mer information om att komma igång med AIR SDK-verktygen finns i Skapa ett AIR-program med verktygen på kommandoraden.

Vad som ingår i AIR SDK

Tabellen nedan beskriver ändamålet med filerna i AIR SDK:

SDK-mapp	Filer/verktygsbeskrivning
bin	Med AIR Debug Launcher (ADL) kan du köra ett AIR-program utan att först paketera och installera det. Information om hur du använder det här verktyget finns i "AIR Debug Launcher (ADL)" på sidan 156. AIR Developer Tool (ADT) paketerar ditt program som en AIR-fil för distribution. Information om hur du använder det här verktyget finns i "AIR Developer Tool (ADT)" på sidan 162.
frameworks	Katalogen libs innehåller kodbibliotek som kan användas i AIR-program. Katalogen projects innehåller koden för de kompilerade SWF- och SWC-biblioteken.
include	Katalogen include innehåller rubrikfilen för C-språket för att skriva ANE-tillägg.
install	Katalogen install innehåller Windows USB-drivrutiner för Android-enheter. (Detta är drivrutinerna från Google som ingår i Android SDK.)
lib	Innehåller stödkod för AIR SDK-verktyg.
runtimes	AIR-miljöerna för stationära datorer och mobilenheter. Skrivbordsmiljön används av ADL för att starta AIR-program innan de har paketerats eller installerats. AIR-miljöer för Android (APK-paket) kan installeras på Android-enheter och -emulatorer för utveckling och testning. Separata APK-paket används för enheter och emulatorer. (Den offentliga AIR-miljön för Android finns på Android Market.)
samples	Den här mappen innehåller ett exempel på en programbeskrivningsfil, ett exempel på den sömlösa installationsfunktionen (badge.swf) och AIR-programmets standardikoner.
templates	descriptor-template.xml - en mall för programbeskrivningsfilen som krävs för varje AIR-program. En detaljerad beskrivning av programbeskrivningsfilen finns i "AIR-programbeskrivningsfiler" på sidan 200. Schemafiler för XML-strukturen i programbeskrivningen för varje officiell version av AIR finns också i den här mappen.

Ställa in Flex SDK

Följande alternativ finns när du vill utveckla Adobe® AIR®-program med Adobe® Flex™:

- Du kan hämta och installera Adobe® Flash® Builder™, som tillhandahåller integrerade verktyg för att skapa Adobe AIR-projekt samt testa, felsöka och paketera AIR-program. Se ”Skapa ditt första Flex AIR-skrivbordsprogram med Flash Builder” på sidan 20
- Du kan hämta Adobe® Flex™ SDK och utveckla Flex AIR-program med den textredigerare och de kommandoradsverktyg som du föredrar.

En snabb genomgång av hur du skapar ett AIR-program med Flex SDK finns i ”Skapa ditt första AIR-skrivbordsprogram med Flex SDK” på sidan 36.

Installera Flex SDK

För att bygga AIR-program med kommandoradsverktygen måste Java vara installerat på datorn. Du kan använda en JVM (Java Virtual Machine) från antingen JRE eller JDK (version 1.5 eller senare). Java JRE och JDK finns på http://java.sun.com/.

Obs! *Java krävs inte för att slutanvändare ska kunna köra AIR-program.*

Flex SDK tillhandahåller AIR API:n och kommandoradsverktyg som du kan använda för att paketera, kompilera och felsöka AIR-program.

1. Om du inte redan har gjort det hämtar du Flex SDK på http://opensource.adobe.com/wiki/display/flexsdk/Downloads.
2. Placera innehållet för SDK i en mapp (t.ex. Flex SDK).
3. Kopiera innehållet i AIR SDK till filerna i Flex SDK.

   Obs! *På Mac måste du se till att du kopierar eller ersätter de enskilda filerna i SDK-mapparna, inte katalogerna. Om du kopierar en katalog på Mac till en katalog med samma namn, tas de befintliga filerna i målkatalogen bort som standard. Innehållet i de båda katalogerna läggs alltså inte samman. Du kan använda kommandot* `ditto` *i ett terminalfönster för att lägga samman AIR SDK med Flex SDK:*`ditto air_sdk_folder flex_sdk_folder`
4. AIR:s kommandoradsverktyg finns i mappen bin.

Konfigurera externa SDK-verktyg

När du utvecklar program för Android och iOS måste du hämta provisioneringsfiler, SDK-verktyg eller andra utvecklingsverktyg från plattformstillverkarna.

Information om hur du hämtar och installerar Android SDK finns i För Android-utvecklare: Installera SDK (på engelska). Från och med AIR 2.6 behöver du inte hämta Android SDK. AIR SDK innehåller nu de grundläggande komponenter som behövs för att installera och starta APK-paket. Men Android SDK kan ändå vara användbart för ett flertal olika utvecklingsuppgifter, bland annat för att skapa och köra programvaruemulatorer samt för att göra skärmdumpar.

Det krävs ingen extern SDK för iOS-utveckling. Det krävs däremot särskilda certifikat och provisioneringsprofiler. Du hittar mer information i Få utvecklarfiler från Apple.

Kapitel 5: Skapa ditt första AIR-program

Skapa ditt första Flex AIR-skrivbordsprogram med Flash Builder

Om du vill ha en snabb och handgriplig illustration av hur Adobe® AIR® fungerar, ska du använda dessa instruktioner för att skapa och paketera ett enkelt SWF-filsbaserat AIR "Hello World"-program med hjälp av Adobe® Flash® Builder.

Hämta och installera Flash Builder om du inte redan gjort det. Hämta och installera även den senaste versionen av Adobe AIR, som finns här: www.adobe.com/go/air_se.

Skapa ett AIR-projekt

I Flash Builder finns verktyg för att utveckla och paketera AIR-program.

Du börjar med att skapa AIR-program i Flash Builder eller Flex Builder på samma sätt som du skapar andra Flex-baserade programprojekt, alltså genom att definiera ett nytt projekt.

1. Öppna Flash Builder.
2. Välj Arkiv > Nytt > Flex-projekt.
3. Ange att projektnamnet ska vara AIRHelloWorld.
4. I Flex tolkas AIR-program som en programtyp. Det finns två alternativ:
   - ett webbprogram som körs i Adobe® Flash® Player
   - ett skrivbordsprogram som körs i Adobe AIR

   Markera Datorprogram som programtyp.
5. Klicka på Slutför för att skapa projektet.

AIR-projekt består av två filer: MXML-huvudfilen och XML-programfilen (kallad programbeskrivningsfilen). Den senare filen anger programegenskaper.

Du hittar mer information i Developing AIR applications with Flash Builder.

Skriva AIR-programkoden

Du skriver koden för "Hello World"-programmet genom att redigera programmets MXML-fil (AIRHelloWorld.mxml), som har öppnats i redigeraren. (Om filen inte är öppen använder du projektnavigeraren för att öppna den.)

Flex AIR-program på skrivbordet finns inuti MXML-taggen WindowedApplication. Med MXML WindowedApplication-taggen skapas ett fönster med enkla fönsterkontroller som exempelvis en namnlist och en stängningsknapp.

1. Lägg till attributet `title` i WindowedApplication-komponenten och tilldela det värdet `"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 Komplettera med en Label-komponent i programmet (placera den i WindowedApplication-taggen). Ange att `text`-egenskapen för Label-komponenten ska vara `"Hello AIR"` och ange att den ska vara centrerad i enlighet med exemplet här:

```
<?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 Lägg till följande formateringsblock omedelbart efter den inledande WindowedApplication-taggen och före taggen för label-komponenten som du angav ovan:

```
<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>
```

Dessa formateringsinställningar gäller för hela programmet och skapar en fönsterbakgrund som är något grå och genomskinlig.

Programkoden ska nu ha följande utseende:

```
<?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>
```

Därefter ska du ändra några inställningar i programbeskrivningen för att göra programmet genomskinligt:

1 I rutan Flex-navigering letar du reda på programbeskrivningsfilen i projektets källkatalog. Om namnet på ditt projekt är AIRHelloWorld, kommer filnamnet att vara AIRHelloWorld-app.xml.

2 Dubbelklicka på programbeskrivningsfilen för att redigera den i Flash Builder.

3 I XML-koden letar du reda på kommentarsraderna för egenskaperna `systemChrome` och `transparent` (i egenskapen `initialWindow`). Ta bort kommentarerna. (Ta bort `"<!--"` and `"-->"` kommentarsavskiljare.)

4 Ange att textvärdet för `systemChrome`-egenskapen ska vara `none` enligt följande exempel:

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

5 Ange att textvärdet för `transparent`-egenskapen ska vara `true` enligt följande exempel:

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

6 Spara filen.

Testa AIR-programmet

Du kan testa programkoden som du skrev genom att köra den i felsökningsläge.

1 Klicka på knappen Felsökning i verktygsfältet.

Du kan även välja Kör > Felsökning > AIRHelloWorld.

Det slutliga AIR-programmet bör se ut som i följande exempel:

2 Med hjälp av egenskaperna `horizontalCenter` och `verticalCenter` i Label-kontrollen placeras texten mitt i fönstret. Flytta eller ändra storlek på fönstret på samma sätt som du arbetar med andra datorprogram.

Obs! *Om programmet inte kompileras måste du ändra syntaxen eller stavningen som du oavsiktligt har lagt till i koden. Fel och varningar visas i problemvyn i Flash Builder.*

Paketera, signera och köra AIR-program

Det är nu dags att börja paketera "Hello World"-programmet i en AIR-fil så att det kan distribueras. En AIR-fil är en arkivfil som innehåller de programfiler som finns i projektets bin-mapp. I detta exempel består dessa filer av SWF-filer och program-XML-filer. Du distribuerar AIR-paketet till användarna som sedan använder det för att installera programmet. En nödvändig åtgärd i sammanhanget är att digitalt signera det.

1. Kontrollera att programmet inte har några kompileringsfel och körs som förväntat.
2. Välj Project (Projekt) > Export Release Build (Exportera färdig version).
3. Kontrollera att AIRHelloWorld-projektet och programmet AIRHelloWorld.mxml listas som projekt och program.
4. Markera alternativet Exportera som signerat AIR-paket. Klicka sedan på Nästa.
5. Om du har ett befintligt digitalt certifikat klickar du på Browse (Bläddra) för att leta reda på och markera det.
6. Markera Skapa om du måste skapa ett nytt självsignerat digitalt certifikat.
7. Ange den obligatoriska informationen och klicka på OK.
8. Klicka på Slutför för att skapa AIR-paketet vilket kommer att få namnet AIRHelloWorld.air.

Du kan nu installera och köra programmet från projektnavigeraren i Flash Builder eller från filsystemet genom att dubbelklicka på AIR-filen.

Skapa ditt första AIR-skrivbordsprogram med Flash Professional

Här får du en snabb, praktisk demonstration av hur Adobe® AIR® fungerar. Följ bara instruktionerna i det här avsnittet så lär du dig att skapa och paketera ett enkelt ”Hello World” AIR-program med Adobe® Flash® Professional.

Om du inte redan har gjort det hämtar och installerar du Adobe AIR, som finns här: www.adobe.com/go/air_se.

Skapa Hello World-programmet i Flash

Att skapa ett Adobe AIR-program i Flash påminner om att skapa en FLA-fil. I exemplet nedan lär du dig att skapa ett enkelt Hello World-program med Flash Professional.

Så här skapar du Hello World-programmet

1. Starta Flash.
2. Klicka på AIR på välkomstskärmen så skapas en tom FLA-fil med publiceringsinställningar för Adobe AIR.
3. Välj textverktyget på verktygspanelen och skapa ett statiskt textfält (standard) mitt på scenen. Gör det tillräckligt brett för att 15–20 tecken ska få plats.
4. Skriv texten ”Hello World” i textfältet.
5. Spara filen och ge den ett namn (till exempel HelloAIR).

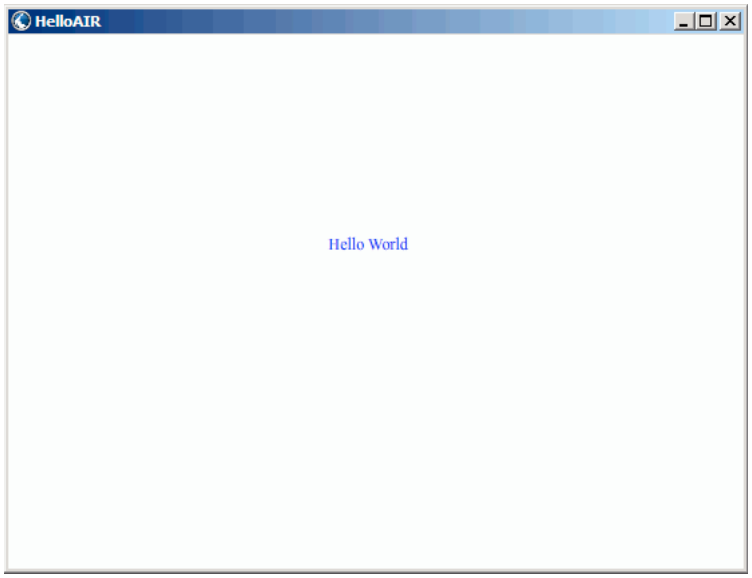
Testa programmet

1. Tryck på Ctrl + Retur eller välj Kontroll -> Testa filmen -> Testa för att testa programmet i Adobe AIR.
2. Om du vill använda funktionen Felsök film läger du först till ActionScript-kod i programmet. Testa det snabbt genom att lägga till en trace-programsats, till exempel:

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

3 Tryck på Ctrl + Skift + Retur eller välj Felsök -> Felsök filmen -> Felsök om du vill köra programmet med Felsök filmen.

Hello World-programmet ser ut som på bilden:



Paketera programmet

1 Välj Arkiv > Publicera.

2 Signera Adobe AIR-paketet med ett befintligt digitalt certifikat eller gör följande och skapa ett självsignerande certifikat:

   a Klicka på knappen Nytt bredvid fältet Certifikat.

   b Fyll i uppgifterna Utgivarnamn, Organisationsenhet, Organisationsnamn, E-post, Land, Lösenord och Bekräfta lösenord.

   c Ange certifikattyp. Alternativet Certifikattyp avser säkerhetsnivån: för 1024-RSA används en 1024-bitars nyckel (mindre säkert) och för 2048-RSA används en 2048-bitars nyckel (säkrare).

   d Spara informationen i en certifikatfil genom att fylla i Spara som eller genom att klicka på Bläddra... och bläddra till en mapp. (Till exempel *C:/Temp/mycert.pfx*). Klicka på OK när du är klar.

   e Flash återvänder till dialogrutan Digital signatur. Sökvägen till och filnamnet för det självsignerade certifikat du skapade visas i textrutan Certifikat. Du kan även ange sökväg och filnamn eller klicka på knappen Bläddra för att hitta och markera det.

   f Ange samma lösenord i lösenordsfältet i dialogrutan Digital signatur som det du angav i steg b. Mer information om signering av Adobe AIR-program finns i ”Signera en AIR-fil digitalt” på sidan 186.

3 Klicka på knappen Publicera för att skapa program- och installationsfilen. (I Flash CS4 och CS5 klickar du på knappen OK.) Du måste köra Testa filmen eller Felsök filmen om du vill skapa SWF-filen och application.xml-filer innan du skapar AIR-filen.

4 Dubbelklicka på AIR-filen (*program*.air) i den mapp där du sparade programmet för att installera det.

5 Klicka på knappen Installera i dialogrutan Programinstallation.

6 Kontrollera inställningar för installationsegenskaper och plats och se till att kryssrutan Starta programmet efter installationen är markerad. Klicka på Fortsätt.

7 Klicka på Slutför när meddelandet Installationen slutförd visas.

Skapa ditt första AIR-program för Android med Flash Professional

Om du vill utveckla AIR-program för Android måste du hämta Flash Professional CS5-tillägget för Android från Adobe Labs.

Du måste också hämta och installera Android SDK från Androids webbplats, enligt beskrivningen i För Android-utvecklare: Installera SDK (på engelska).

Skapa ett projekt

1 Öppna Flash Professional CS5

2 Skapa ett nytt AIR for Android-projekt.

På hemskärmen i Flash Professional finns en länk för att skapa ett AIR for Android-program. Du kan också välja Arkiv > Nytt och välja AIR for Android-mallen.

3 Spara dokumentet som HelloWorld.fla

Skriv koden

Eftersom den här självstudien inte handlar om hur du skriver kod använder du bara textverktyget för att skriva "Hello, World!" på scenen.

Ange programegenskaper

1 Välj Arkiv > AIR for Android-inställningar.

2 Ange följande inställningar på fliken Allmänt:

- Utdatafil: HelloWorld.apk
- Programnamn: HelloWorld
- Program-ID: HelloWorld
- Versionsnummer: 0.0.1
- Proportioner: Stående

3 Ange följande inställningar på fliken Distribution:

- Certifikat: Peka på ett giltigt AIR-kodsigneringscertifikat. Du kan klicka på knappen Skapa om du vill skapa ett nytt certifikat. (Android-program som distribueras via Android Marketplace måste ha certifikat som är giltiga till minst år 2033.) Ange lösenordet för certifikatet i fältet Lösenord.
- Typ av Android-distribution: Felsök
- Efter publicering: Markera båda alternativen
- Ange sökvägen till ADB-verktyget i underkatalogen tools i Android SDK.

4 Stäng dialogrutan med inställningar för Android genom att klicka på OK.

Programmet behöver inte ikoner eller behörigheter i den här utvecklingsfasen. De flesta AIR-program för Android behöver vissa behörigheter för att komma åt skyddade funktioner. Du bör bara ange de behörigheter programmet verkligen har behov av, eftersom användare kan låta bli att använda ditt program om det kräver för många behörigheter.

5 Spara filen.

Paketera och installera programmet på Android-enheten

1 Kontrollera att USB-felsökning är aktiverat på enheten. Du kan aktivera USB-felsökning i inställningarna under Applications > Development.

2 Anslut enheten till datorn med en USB-kabel.

3 Installera AIR-miljön, om du inte redan har gjort det, genom att besöka Android Market och hämta Adobe AIR. (Du kan också installera AIR lokalt med ”ADT-kommandot installRuntime” på sidan 174. Android-paket för användning på Android-enheter och -emulatorer ingår i AIR SDK.)

4 Välj Arkiv > Publicera.

APK-filen skapas i Flash Professional, och programmet installeras på den anslutna Android-enheten och startas.

Skapa ditt första AIR-program för iOS

AIR 2.6 eller senare, iOS 4.2 eller senare

Du kan koda, bygga och testa de grundläggande funktionerna i ett iOS-program med enbart Adobe-verktyg. Men om du vill installera ett iOS-program på en enhet och distribuera det programmet måste du gå med i Apple iOS Developer Program (en kostnadsbelagd tjänst). När du väl är med i Apple iOS Developer Program får du tillgång till iOS Provisioning Portal, där du kan hämta de objekt och filer (från Apple) som behövs för att installera ett program på en enhet för testning och efterföljande distribution. Följande objekt och filer behövs:

- Certifikat för utveckling och distribution
- Program-ID:n
- Provisioneringsfiler för utveckling och distribution

Skapa programinnehållet

Skapa en SWF-fil som visar texten ”Hello world!”. Du kan göra detta med Flash Professional, Flash Builder eller en annan integrerad utvecklingsmiljö. I det här exemplet används en textredigerare och det kommandoradsverktyg för SWF-kompilering som ingår i Flex SDK.

1 Skapa en katalog på en lämplig plats för att spara programfilerna. Skapa en fil med namnet *HelloWorld.as* och redigera filen i önskat kodredigeringsprogram.

2 Lägg till följande kod:

```
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 Kompilera klassen med amxmlc-kompileraren:

```
amxmlc HelloWorld.as
```

En SWF-fil med namnet *HelloWorld.swf* skapas i samma mapp.

Obs! *Det här exemplet förutsätter att du har konfigurerat systemvariabeln path så att den katalog som innehåller amxmlc finns med. Information om hur du anger sökvägen finns i ”Systemvariabeln path” på sidan 299. Du kan också skriva den fullständiga sökvägen till amxmlc och de andra kommandoradsverktygen som används i det här exemplet.*

Skapa ikoner och inledande skärmbilder för programmet

Alla iOS-program innehåller ikoner som visas i användargränssnittet i iTunes-programmet och på enhetens skärm.

1 Skapa en katalog i din projektkatalog med namnet ”icons”.

2 Skapa tre PNG-filer i katalogen ”icons”. Ge dem namnen Icon_29.png, Icon_57.png och Icon_512.png.

3 Redigera PNG-filerna till den typ av grafik som du vill ha för ditt program. Filerna måste vara 29 x 29 pixlar, 57 x 57 pixlar och 512 x 512 pixlar. För det här testet kan du helt enkelt använda enfärgade fyrkanter som grafik.

Obs! *När du skickar ett program till Apples App Store använder du en JPG-version (inte en PNG-version) av filen med 512 pixlar. PNG-versionen använder du medan du testar utvecklingsversionerna av ett program.*

För alla iPhone-program visas en inledande bild när programmet läses in på iPhone-enheten. Du skapar den inledande bilden i en PNG-fil:

1 Skapa en PNG-fil med namnet Default.png i utvecklingens huvudkatalog. (Lägg *inte* den här filen i underkatalogen ”icons”. Se till att du ger filen namnet Default.png, med ett versalt D.)

2 Redigera filen så att den är 320 pixlar bred och 480 pixlar hög. Tills vidare kan innehållet vara en tom vit rektangel. (Du kommer att ändra detta senare.)

Detaljerad information om de här bilderna finns i ”Programikoner” på sidan 85.

Skapa programbeskrivningsfilen

Skapa en programbeskrivningsfil som anger programmets grundläggande egenskaper. Du kan göra detta med en integrerad utvecklingsmiljö som Flash Builder eller med en textredigerare.

1 Skapa en XML-fil med namnet *HelloWorld-app.xml* i den projektmapp som innehåller filen HelloWorld.as. Redigera den här filen i önskat XML-redigeringsprogram.

2 Lägg till följande XML-kod:

```
<?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>
```

För enkelhets skull anges bara ett fåtal av de tillgängliga egenskaperna i det här exemplet.

Obs! *Om du använder AIR 2, eller tidigare, måste du använda elementet* `<version>` *i stället för elementet* `<versionNumber>`.

3 Ändra program-ID:t så att det matchar det program-ID som anges i iOS Provisioning Portal. (Inkludera inte den slumpmässiga källpaketdelen i början av ID:t.)

4 Testa programmet med ADL:

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

ADL bör öppna ett fönster på skrivbordet med texten *Hello World!*. Om inget fönster öppnas kontrollerar du om det finns några fel i källkoden eller programbeskrivningen.

Kompilera IPA-filen

Du kan nu kompilera IPA-installationsfilen med ADT. Du måste ha ditt utvecklarcertifikat från Apple och din privata nyckel i P12-filformat samt din provisioneringsprofil för utveckling från Apple.

Kör ADT-verktyget med följande alternativ och ersätt värdena för keystore, storepass och provisioning-profile med dina värden:

```
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
```

(Använd bara en kommandorad. Radbrytningarna i exemplet är bara för att det ska vara lättare att läsa.)

ADT genererar installationsfilen för iOS-programmet, *HelloWorld.ipa*, i projektkatalogen. Det kan ta några minuter att kompilera IPA-filen.

Installera programmet på en enhet

Så här installerar du iOS-programmet för testning:

1 Öppna iTunes-programmet.

2 Om du inte redan har gjort det ska du lägga till provisioneringsprofilen för programmet i iTunes. Välj Arkiv > Lägg till i biblioteket i iTunes. Välj sedan filen med provisioneringsprofilen (med filtypen mobileprovision).

   När du nu testar programmet på din utvecklarenhet använder du provisioneringsprofilen för utveckling.

   Sedan när du distribuerar programmet till iTunes Store använder du profilen för distribution. Om du vill göra en ad hoc-distribution av programmet (till flera enheter utan att gå via iTunes Store) använder du ad hoc-provisioneringsprofilen.

   Mer information om provisioneringsprofiler finns i ”iOS-konfiguration” på sidan 64.

3 I vissa versioner av iTunes ersätts inte programmet om samma version av programmet redan finns installerat. I så fall tar du bort programmet från din enhet och från listan över program i iTunes.

4 Dubbelklicka på IPA-filen för programmet. Den bör visas i listan med program på iTunes.

5 Anslut enheten till USB-porten på datorn.

6 Kontrollera programfliken för enheten i iTunes och att programmet är markerat i listan över program som ska installeras.

7 Markera enheten i den vänstra listan i iTunes-programmet. Klicka sedan på Synkronisera. När synkroniseringen är slutförd visas programmet Hello World i din iPhone.

Om den nya versionen inte har installerats tar du bort programmet från enheten och från listan över program i iTunes och upprepar sedan proceduren. Detta kan bero på att den installerade versionen använder samma program-ID och version.

Redigera grafiken för den inledande skärmen

Innan du kompilerade programmet skapade du filen Default.png (se ”Skapa ikoner och inledande skärmbilder för programmet” på sidan 27). Den här PNG-filen visas som startbild medan programmet läses in. När du testade programmet på din iPhone kanske du lade märke till den tomma skärmen som visades när programmet startades.

Du bör ändra den här bilden för att matcha startskärmen för programmet (”Hello World!”):

1 Öppna programmet på din enhet. När den första Hello World-texten visas trycker du på och håller ned hemknappen (under skärmen). Samtidigt som du håller ned hemknappen trycker du på viloläge-/strömknappen (på ovansidan av din iPhone). Då tas en skärmavbild som skickas till kamerarullen.

2 Överför bilden till utvecklingsdatorn med iPhoto eller med något annat bildöverföringsprogram. (I Mac OS kan du också använda programmet Image Capture.)

   Du kan också skicka bilden via e-post till utvecklingsdatorn:

   - Öppna bildprogrammet.
   - Öppna kamerarullen.
   - Öppna den skärmavbild som du har tagit.

- Tryck på bilden och tryck sedan på framåtpilen längst ned till vänster. Klicka sedan på knappen E-posta bild och skicka bilden till dig själv.

3 Ersätt filen Default.png (i utvecklingskatalogen) med en PNG-version av skärmavbilden.

4 Kompilera om programmet (se ”Kompilera IPA-filen” på sidan 28) och installera om det på enheten.

Den nya startskärmen används nu under det att programmet läses in.

Obs! *Du kan skapa vilken grafik du vill i filen Default.png, så länge som grafiken har rätt dimensioner (320 x 480 pixlar). Men det är oftast bäst att bilden Default.png matchar det inledande tillståndet för programmet.*

Skapa ditt första HTML-baserade AIR-program med Dreamweaver

Om du vill få en illustration av hur Adobe® AIR® fungerar ska du använda dessa instruktioner för att skapa och paketera ett enkelt HTML-baserat AIR "Hello World"-program med hjälp av Adobe® AIR®-tillägget för Dreamweaver®.

Om du inte redan har gjort det hämtar och installerar du Adobe AIR, som finns här: www.adobe.com/go/air_se.

Information om hur du installerar Adobe AIR-tillägget för Dreamweaver finns i Installera AIR-tillägget för Dreamweaver.

En översikt över tillägget, inklusive systemkrav finns i AIR-tillägget för Dreamweaver.

Obs! *HTML-baserade AIR-program kan bara utvecklas för profilerna desktop och extendedDesktop. Mobilprofilen stöds inte.*

Förbereda programfiler

Ditt Adobe AIR-program måste ha en startsida och alla relaterade sidor definierade på en Dreamweaver-plats:

1 Starta Dreamweaver och se till att det finns en plats definierad.

2 Öppna en HTML-sida genom att välja Arkiv > Nytt, markera HTML i kolumnen Sidtyp, markera Inga i kolumnen Layout och klicka sedan på Skapa.

3 På den nya sidan skriver du **Hello World!**

Detta exempel är mycket enkelt, men om du vill kan du formatera texten, lägga till mer innehåll på sidan, länka andra sidor till denna startsida och mycket mera.

4 Spara sidan (Arkiv > Spara) som hello_world.html. Se till att du sparar filen på en Dreamweaver-plats.

Mer information om Dreamweaver-platser finns i Dreamweaver-hjälp.

Skapa Adobe AIR-programmet

1 Kontrollera att sidan hello_world.html är öppen i dokumentfönstret i Dreamweaver. (I föregående avsnitt beskrivs hur du skapar den.)

2 Välj Plats > Inställningar för AIR-program.

De flesta obligatoriska inställningarna i dialogrutan Dialogrutan AIR – program- och installationsinställningar fylls i automatiskt. Du måste emellertid välja det ursprungliga innehållet (eller startsidan) för programmet.

3 Klicka på knappen Bläddra intill alternativet Ursprungligt innehåll och navigera till sidan hello_world.html för att markera den.

4 Intill alternativet Digital signatur klickar du på knappen Ange.

En digital signatur säkerställer att koden för ett program inte har ändrats eller förstörts sedan den skapades av programutvecklaren och den krävs dessutom för alla Adobe AIR-program.

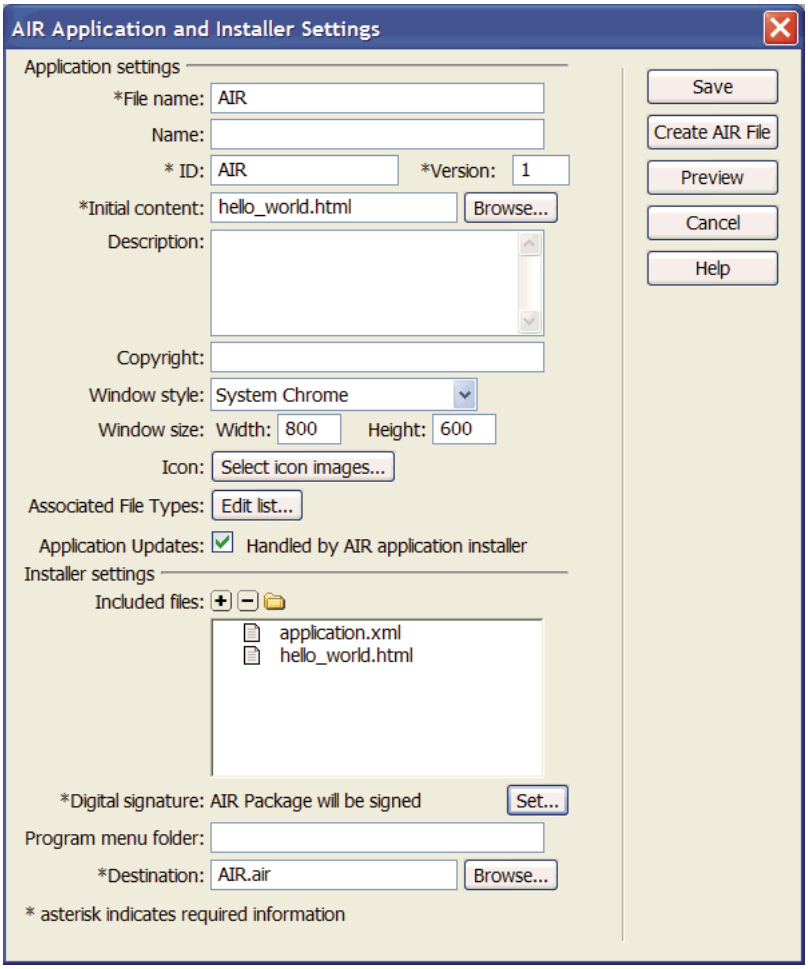
5 I dialogrutan Digital signatur väljer du Signera AIR-paketet med ett digitalt certifikat och klickar sedan på knappen Skapa. (Om du redan har tillgång till ett digitalt certifikat kan du i stället klicka på knappen Bläddra och markera det.)

6 Fyll i de obligatoriska fälten i dialogrutan Självsignerat digitalt certifikat. Du måste ange ditt namn och ett lösenord och bekräfta det. Ange därefter ett namn för filen för det digitala certifikatet. I Dreamweaver sparas det digitala certifikatet i platsroten.

7 Klicka på OK för att gå tillbaka till dialogrutan Digital signatur.

8 I dialogrutan Digital signatur anger du lösenordet som du angav för det digitala certifikatet och klickar sedan på OK.

Dialogrutan AIR – program- och installationsinställningar kommer nu att se ut ungefär så här:



Mer detaljerad information om alla alternativ i dialogrutorna och hur du ändrar dem finns i Skapa ett AIR-program i Dreamweaver.

9 Klicka på knappen Skapa AIR-fil.

I Dreamweaver skapas Adobe AIR-programfilen som sparas i platsrotmappen. Dessutom skapas i Dreamweaver filen application.xml som även den sparas på samma plats. Filen fungerar som ett manifest för definition av olika programegenskaper.

Installera program på en dator

Nu när du har skapat en programfil kan du installera den på en dator.

1 Flytta Adobe AIR-programfilen bort från Dreamweaver-platsen till ditt eller något annat skrivbord.

Detta är valfritt. Du kan faktiskt installera det nya programmet på din dator direkt från Dreamweavers platskatalog.

2 Dubbelklicka på den körbara programfilen (.air) för att installera programmet.

Förhandsvisa Adobe AIR-programmet

Du kan när som helst förhandsvisa sidor som kommer att ingå som delar av AIR-program. Det är alltså inte nödvändigt att paketera programmet innan du kan se hur det kommer att se ut när det är installerat.

1 Kontrollera att sidan hello_world.html är öppen i Dreamweavers dokumentfönster.

2 I verktygsfältet Dokument klickar du på knappen Förhandsgranska/felsök i webbläsare och väljer sedan Förhandsvisa i AIR.

Du kan även trycka på Ctrl+Skift+F12 (Windows) eller Cmd+Skift+F12 (Macintosh).

När du förhandsvisar den här sidan ser du de facto det som användaren kommer att se som startsida för programmet efter det att han/hon installerat programmet.

Skapa ditt första HTML-baserade AIR-program med AIR SDK

Om du vill få en illustration av hur Adobe® AIR® fungerar ska du använda dessa instruktioner för att skapa och paketera ett enkelt HTML-baserat AIR "Hello World"-program.

Till att börja med måste du installera runtime-modulen och ställa in AIR SDK. Du kommer att använda *AIR Debug Launcher* (ADL) och *AIR Developer Tool* (ADT) i denna självstudie. ADL och ADT är kommandoradsverktyg och du hittar dem i katalogen `bin` för AIR SDK (se ”Installera AIR SDK” på sidan 17). I denna självstudie antas att du redan känner till hur du kör program från kommandoraden och vet hur man ställer in nödvändiga sökvägsmiljövariabler för det operativsystem som du arbetar med.

Obs! *Om du använder Adobe® Dreamweaver® läser du ”Skapa ditt första HTML-baserade AIR-program med Dreamweaver” på sidan 30.*

Obs! *HTML-baserade AIR-program kan bara utvecklas för profilerna desktop och extendedDesktop. Mobilprofilen stöds inte.*

Skapa projektfiler

Alla HTML-baserade AIR-projekt måste innehålla två filer; en programbeskrivningsfil, för att ange applikationens metadata, och en HTML-sida på översta nivån. Förutom dessa obligatoriska filer innehåller detta projekt en JavaScript-fil, `AIRAliases.js`, som definierar användbara aliasvariabler för AIR API-klasserna.

1 Skapa en katalog med namnet `HelloWorld` för projektfilerna.

2 Skapa en XML-fil med namnet `HelloWorld-app.xml`.

3 Skapa en HTML-fil med namnet `HelloWorld.html`.

4 Kopiera `AIRAliases.js` från ramverkets AIR SDK-mapp till projektkatalogen.

Skapa programbeskrivningsfilen för AIR

Börja med att skapa en XML-programbeskrivningsfil med följande utseende när du vill bygga upp ett AIR-program.

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

1 Öppna filen `HelloWorld-app.xml` för redigering.

2 Lägg till rotelementet `<application>` inklusive AIR-namnutrymmesattributet:

<application xmlns="http://ns.adobe.com/air/application/2.7"> Det sista segmentet i namnutrymmet, "2.7", anger vilken version av miljön som krävs för att köra programmet.

3 Lägg till elementet `<id>`:

<id>examples.html.HelloWorld</id> Program-ID tillsammans med utgivar-ID identifierar programmet (som härleds i AIR från certifikatet som används för att signera programpaketet). Program-ID används för installationer, åtkomst till kataloger i privata filsystem, åtkomst till privata krypterade lagringsutrymmen och för kommunikation mellan program.

4 Lägg till elementet `<versionNumber>`:

<versionNumber>0.1</versionNumber> Detta hjälper användarna att avgöra vilken version av ditt program de installerar.

Obs! *Om du använder AIR 2, eller tidigare, måste du använda elementet `<version>` i stället för elementet `<versionNumber>`.*

5 Lägg till elementet `<filename>`:

<filename>HelloWorld</filename> Det namn som används för den körbara programfilen, installationskatalogen och andra referenser i operativsystemet till programmet.

6 Lägg till elementet `<initialWindow>` som innehåller följande underordnade element för att ange egenskaperna för det första programfönstret:

<content>HelloWorld.html</content> identifierar HTML-rotfilen som ska läsas in i AIR.

<visible>true</visible> gör att fönstret visas omedelbart.

<width>400</width> anger fönstrets bredd i pixlar.

<height>200</height> anger fönstrets höjd.

7 Spara filen. Den slutgiltiga programbeskrivningsfilen ska nu ha följande utseende:

```
<?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>
```

I detta exempel ställs endast ett fåtal av de möjliga programegenskaperna in. Mer information om alla programegenskapar, som du använder för att ange exempelvis fönsterkontroller, fönsterstorlek, genomskinlighet, standarkatalog för installation, associerade filtyper och programikoner, finns i ”AIR-programbeskrivningsfiler” på sidan 200.

Skapa HTML-sidan för programmet

Du måste nu skapa en enkel HTML-sida som ska användas som huvudfil för AIR-programmet.

1 Öppna filen `HelloWorld.html` för redigering. Skriv in följande HTML-kod:

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

2 I HTLM-avsnittet `<head>` importerar du filen `AIRAliases.js`:

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

I AIR definieras en egenskap med namnet `runtime` i HTML-fönsterobjektet. Genom runtime-egenskapen får du åtkomst till de interna AIR-klasserna genom att använda det fullständiga namnet för klassen. Om du till exempel vill skapa ett AIR File-objekt kan du lägga till följande JavaScript-sats:

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

Filen `AIRAliases.js` definierar alias för de mest användbara AIR API:erna. Genom att använda `AIRAliases.js` kan du korta ner referensen till File-klassen enligt följande:

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

3 Under script-taggen AIRAliases lägger du till ytterligare en script-tagg som ska innehålla en JavaScript-funktion för att hantera händelsen `onLoad`:

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

Funktionen `appLoad()` används endast för att anropa funktionen `air.trace()`. Trace-meddelandet skrivs ut på kommandokonsolen när du kör programmet med ADL. Dessa trace-satser kan vara mycket användbara vid felsökning.

4 Spara filen.

Filen `HelloWorld.html` ske nu ha följande utseende:

```
<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>
```

Testa programmet

Använd ADL-verktyget (AIR Debug) för att köra och testa programmet från kommandoraden. ADL-filen finns i katalogen `bin` i AIR SDK-mappen. Om du inte redan installerat AIR SDK ska du läsa ”Installera AIR SDK” på sidan 17.

1 Öppna en kommandokonsol eller ett skal. Ändra till katalogen som du skapade för detta projekt.

2 Kör följande kommando:

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

Ett AIR-fönster öppnas och ditt program visas. I konsolfönstret visas meddelandet för `air.trace()`-anropet.

Du hittar mer information i ”AIR-programbeskrivningsfiler” på sidan 200.

Skapa AIR-installationsfilen

När ditt program fungerar som det ska kan du använda ADT-verktyget för att paketera programmet i en AIR-installationsfil. En AIR-installationsfil är en arkivfil som innehåller alla programfiler och den kan distribueras till användarna. Du måste installera Adobe AIR innan du kan installera en paketerad AIR-fil.

Om du vill behålla programsäkerheten måste alla AIR-installationsfiler först signeras digitalt. Du kan underlätta utvecklingen genom att skapa ett grundläggande självsignerat certifikat med ADT eller något annat certifikatgenereringsverktyg. Du kan även köpa ett kommersiellt kodsigneringscertifikat från företag som VeriSign eller Thawte. När användarna installerar en självsignerad AIR-fil, visas utgivaren som "okänd" under installationsprocessen. Detta beror på att självsignerade certifikat endast garanterar att AIR-filen inte har ändrats sedan den skapades. Det finns inget som hindrar någon från att självsignera en maskerad AIR-fil och presenterar den som sin egen. Om det avser offentliga AIR-filer rekommenderar vi ett verifieringsbart kommersiellt certifikat. En översikt över säkerhetsfrågor i AIR finns i AIR-säkerhet (för ActionScript-utvecklare) eller AIR security (för HTML-utvecklare).

Generera ett självsignerat certifikat och nyckelpar

❖ I kommandotolken anger du följande kommando (den körbara ADT-filen finns i katalogen `bin` i AIR SDK):

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

I ADT genereras en nyckelbehållarfil med namnet *sampleCert.pfx* vilken innehåller ett certifikat och den aktuella privata nyckeln.

Detta exempel innehåller det minsta antal attribut som kan anges för ett certifikat. Nyckeltypen måste vara antingen *1024-RSA* eller *2048-RSA* (läs mer i ”Signera AIR-program” på sidan 186).

Skapa AIR-installationsfilen

❖ I kommandotolken skriver du följande kommando (på en rad):

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

Du kommer att få en fråga om nyckelfilens lösenord.

HelloWorld.air-argumentet är AIR-filen som skapas i ADT. HelloWorld-app.xml är programbeskrivningsfilen. De efterföljande argumenten är filer som används i ditt program. I detta exempel används endast två filer, men du kan ta med det antal filer och kataloger som du anser dig behöva. ADT verifierar att den huvudsakliga innehållsfilen, HelloWorld.html, ingår i paketet, men om du glömmer att inkludera AIRAliases.js kommer ditt program inte att fungera.

När AIR-paketet har skapats kan du installera och köra programmet genom att dubbelklicka på paketfilen. Du kan också skriva AIR-filnamnet som ett kommando i ett skal eller i kommandofönstret.

Nästa steg

I AIR fungerar HTML- och JavaScript-kod vanligtvis på samma sätt som i en vanlig webbläsare. (Faktum är att det är samma WebKit-återgivningsmotor som i webbläsaren Safari.) Det finns emellertid några viktiga skillnader som du måste vara medveten om när du utvecklar HTML-program i AIR. Mer information om skillnaderna samt andra viktiga ämnen finns i Programming HTML and JavaScript.

Skapa ditt första AIR-skrivbordsprogram med Flex SDK

Om du vill ha en snabb och tydlig illustration av hur Adobe® AIR® fungerar kan du använda de här instruktionerna för att skapa ett enkelt SWF-baserat "Hello World"-program i AIR med hjälp av Flex SDK. Den här självstudien visar hur du kompilerar, testar och paketerar ett AIR-program med de kommandoradsverktyg som ingår i Flex SDK (AIR SDK ingår i Flex SDK).

Till att börja med måste du installera runtime-modulen och ställa in Adobe® Flex™. I den här självstudien används kompilatorn *AMXMLC*, *AIR Debug Launcher* (ADL) och *AIR Developer Tool* (ADT). De här programmen finns i katalogen `bin` i Flex SDK (se ”Ställa in Flex SDK” på sidan 19).

Skapa programbeskrivningsfilen för AIR

I det här avsnittet beskrivs hur du skapar programbeskrivningen, som är en XML-fil med följande struktur:

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

1 Skapa en XML-fil med namnet `HelloWorld-app.xml` och spara den i projektkatalogen.

2 Lägg till elementet `<application>`, inklusive AIR-namnutrymmesattributet:

<application xmlns="http://ns.adobe.com/air/application/2.7"> Det sista segmentet i namnutrymmet, "2.7", anger vilken version av miljön som krävs för att köra programmet.

3 Lägg till elementet `<id>`:

<id>samples.flex.HelloWorld</id> Program-ID tillsammans med utgivar-ID identifierar programmet (som härleds i AIR från certifikatet som används för att signera programpaketet). Det rekommenderade formatet är en punktavgränsad sträng med "omvänd DNS", till exempel `"se.foretag.Programnamn"`. Program-ID används för installationer, åtkomst till kataloger i privata filsystem, åtkomst till privata krypterade lagringsutrymmen och för kommunikation mellan program.

4 Lägg till elementet `<versionNumber>`:

<versionNumber>1.0</versionNumber> Detta hjälper användarna att avgöra vilken version av ditt program de installerar.

Obs! *Om du använder AIR 2, eller tidigare, måste du använda elementet `<version>` i stället för elementet `<versionNumber>`.*

5 Lägg till elementet `<filename>`:

<filename>HelloWorld</filename> Det namn som används för den körbara programfilen, installationskatalogen och liknande för referenser i operativsystemet.

6 Lägg till elementet `<initialWindow>` som innehåller följande underordnade element för att ange egenskaperna för det första programfönstret:

<content>HelloWorld.swf</content> Identifierar SWF-rotfilen som ska läsas in i AIR.

<visible>true</visible> gör att fönstret visas omedelbart.

<width>400</width> anger fönstrets bredd i pixlar.

<height>200</height> anger fönstrets höjd.

7 Spara filen. Den fullständiga programbeskrivningsfilen bör se ut så här:

```
<?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>
```

I detta exempel ställs endast ett fåtal av de möjliga programegenskaperna in. Mer information om alla programegenskapar, som du använder för att ange exempelvis fönsterkontroller, fönsterstorlek, genomskinlighet, standarkatalog för installation, associerade filtyper och programikoner, finns i ”AIR-programbeskrivningsfiler” på sidan 200.

Skriva programkoden

Obs! *SWF-baserade AIR-program kan använda en huvudklass som definieras antingen med MXML eller med Adobe® ActionScript® 3.0. I det här exemplet används en MXML-fil för att definiera dess huvudklass. Ett AIR-program med en ActionScript-huvudklass skapas på ungefär samma sätt. I stället för att kompilera en MXML-fil till SWF-filen, kompilerar du ActionScript-klassfilen. När du använder ActionScript måste huvudklassen utöka flash.display.Sprite.*

Precis som alla Flex-baserade program innehåller AIR-program byggda med Flex-ramverket en MXML-huvudfil. I AIR-skrivbordsprogram används emellertid komponenten *WindowedApplication* som rotelement i stället för Application-komponenten. Komponenten WindowedApplication innehåller egenskaper, metoder och händelser för att styra programmet och dess första fönster. På plattformar och i profiler för vilka flera fönster inte stöds i AIR fortsätter du att använda Application-komponenten. I Flex-program för mobiler kan du också använda komponenterna View och TabbedViewNavigatorApplication.

Så här skapar du Hello World-programmet:

1. Med hjälp av en textredigerare skapar du en fil med namnet `HelloWorld.mxml` och lägger till följande MXML-kod:

```
<?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. Sedan lägger du till en Label-komponent i programmet (placera den inuti WindowedApplication-taggen).
3. Ställ in egenskapen `text` för Label-komponenten på `"Hello AIR"`.
4. Ställ in layoutbegränsningarna så att den alltid är centrerad.

   Följande exempel visar koden så här långt:

```
<?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>
```

Kompilera programmet

Innan du kan köra och felsöka programmet kompilerar du MXML-koden till en SWF-fil med hjälp av amxmlc-kompilatorn. amxmlc-kompilatorn finns i katalogen `bin` i Flex SDK. Om du vill kan du ställa in datorns sökvägsmiljö så att den omfattar bin-katalogen i Flex SDK. Om du ställer in sökvägen blir det enklare att köra verktygen på kommandoraden.

1 Öppna ett kommandoskal eller en terminal och navigera till projektmappen för AIR-programmet.

2 Ange följande kommando:

```
amxmlc HelloWorld.mxml
```

Om du kör `amxmlc` skapas `HelloWorld.swf`, som innehåller den kompilerade koden i programmet.

Obs! *Om programmet inte kompileras åtgärdar du syntax- eller stavfel. Fel och varningar visas i konsolfönstret som används för att köra amxmlc-kompilatorn.*

Mer information finns i ”Kompilera MXML- och ActionScript-källfiler för AIR” på sidan 152.

Testa programmet

Om du vill köra och testa programmet från kommandoraden använder du AIR Debug Launcher (ADL) för att starta programmet med hjälp av dess programbeskrivningsfil. (ADL finns i katalogen bin i Flex SDK.)

❖ I kommandotolken skriver du följande kommando:

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

Det resulterande AIR-programmet ser ut ungefär så här:

Med hjälp av egenskaperna horizontalCenter och verticalCenter i Label-kontrollen placeras texten mitt i fönstret. Flytta eller ändra storlek på fönstret på samma sätt som du arbetar med andra datorprogram.

Du hittar mer information i ”AIR Debug Launcher (ADL)” på sidan 156.

Skapa AIR-installationsfilen

När ditt program fungerar som det ska kan du använda ADT-verktyget för att paketera programmet i en AIR-installationsfil. En AIR-installationsfil är en arkivfil som innehåller alla programfiler och den kan distribueras till användarna. Du måste installera Adobe AIR innan du kan installera en paketerad AIR-fil.

Om du vill behålla programsäkerheten måste alla AIR-installationsfiler först signeras digitalt. Du kan underlätta utvecklingen genom att skapa ett grundläggande självsignerat certifikat med ADT eller något annat certifikatgenereringsverktyg. Du kan även köpa ett kommersiellt kodsigneringscertifikat från en kommersiell certifikatutfärdare. När användarna installerar en självsignerad AIR-fil, visas utgivaren som "okänd" under installationsprocessen. Detta beror på att självsignerade certifikat endast garanterar att AIR-filen inte har ändrats sedan den skapades. Det finns inget som hindrar någon från att självsignera en maskerad AIR-fil och presenterar den som sin egen. Om det avser offentliga AIR-filer rekommenderar vi ett verifieringsbart kommersiellt certifikat. En översikt över säkerhetsfrågor i AIR finns i AIR-säkerhet (för ActionScript-utvecklare) eller AIR security (för HTML-utvecklare).

Generera ett självsignerat certifikat och nyckelpar

❖ I kommandotolken anger du följande kommando (den körbara ADT-filen finns i katalogen `bin` i Flex SDK):

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

Detta exempel innehåller det minsta antal attribut som kan anges för ett certifikat. Nyckeltypen måste vara antingen *1024-RSA* eller *2048-RSA* (läs mer i ”Signera AIR-program” på sidan 186).

Skapa AIR-paketet

❖ I kommandotolken skriver du följande kommando (på en rad):

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

Du kommer att få en fråga om nyckelfilens lösenord. Skriv lösenordet och tryck på Retur. Tecknen i lösenordet visas inte av säkerhetsskäl.

HelloWorld.air-argumentet är AIR-filen som skapas i ADT. HelloWorld-app.xml är programbeskrivningsfilen. De efterföljande argumenten är filer som används i ditt program. I detta exempel används endast tre filer, men du kan ta med det antal filer och kataloger som du anser dig behöva.

När AIR-paketet har skapats kan du installera och köra programmet genom att dubbelklicka på paketfilen. Du kan också skriva AIR-filnamnet som ett kommando i ett skal eller i kommandofönstret.

Mer information finns i ”Paketera en AIR-installationsfil för skrivbordet” på sidan 52.

Skapa ditt första AIR-program för Android med Flex SDK

Innan du börjar måste du installera och konfigurera AIR SDK och Flex SDK. I den här självstudien används *AMXMLC*-kompileraren från Flex SDK och *AIR Debug Launcher* (ADL) samt *AIR Developer Tool* (ADT) från AIR SDK. Läs mer i ”Ställa in Flex SDK” på sidan 19.

Du måste också hämta och installera Android SDK från Androids webbplats, enligt beskrivningen i För Android-utvecklare: Installera SDK (på engelska).

Obs! *Information om iPhone-utveckling finns i Skapa ett ”Hello World” iPhone-program med Flash Professional CS5.*

Skapa programbeskrivningsfilen för AIR

I det här avsnittet beskrivs hur du skapar programbeskrivningen, som är en XML-fil med följande struktur:

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

1 Skapa en XML-fil med namnet `HelloWorld-app.xml` och spara den i projektkatalogen.

2 Lägg till elementet `<application>`, inklusive AIR-namnutrymmesattributet:

<application xmlns="http://ns.adobe.com/air/application/2.7"> Det sista segmentet i namnutrymmet, "2.7", anger vilken version av miljön som krävs för att köra programmet.

3 Lägg till elementet `<id>`:

<id>samples.android.HelloWorld</id> Program-ID tillsammans med utgivar-ID identifierar programmet (som i AIR härleds från det certifikat som användes för att signera programpaketet). Det rekommenderade formatet är en punktavgränsad sträng med omvänd DNS-notation, till exempel `"se.foretag.Programnamn"`.

4 Lägg till elementet `<versionNumber>`:

<versionNumber>0.0.1</versionNumber> Detta hjälper användarna att avgöra vilken version av ditt program de installerar.

5 Lägg till elementet `<filename>`:

<filename>HelloWorld</filename> Det namn som används för den körbara programfilen, installationskatalogen och liknande för referenser i operativsystemet.

6 Lägg till elementet `<initialWindow>` som innehåller följande underordnade element för att ange egenskaperna för det första programfönstret:

<content>HelloWorld.swf</content> Identifierar HTML-rotfilen som ska läsas in i AIR.

7 Lägg till elementet `<supportedProfiles>`.

<supportedProfiles>mobileDevice</supportedProfiles> Anger att programmet bara körs i mobilprofilen.

8 Spara filen. Den fullständiga programbeskrivningsfilen bör se ut så här:

```
<?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>
```

I detta exempel ställs endast ett fåtal av de möjliga programegenskaperna in. Det finns även andra inställningar som du kan använda i programbeskrivningsfilen. Du kan till exempel lägga till <fullScreen>true</fullScreen> i elementet initialWindow för att skapa ett helskärmsprogram. Om du vill aktivera fjärrfelsökning och skyddade funktioner på Android måste du också lägga till Android-behörigheter i programbeskrivningen. Det här enkla programmet behöver inga behörigheter, så du behöver inte lägga till dem nu.

Mer information finns i ”Ange egenskaper för mobilprogram” på sidan 69.

Skriva programkoden

Skapa en fil med namnet HelloWorld.as och lägg till följande kod med en textredigerare:

```
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 );
        }
    }
}
```

Kompilera programmet

Innan du kan köra och felsöka programmet kompilerar du MXML-koden till en SWF-fil med hjälp av amxmlc-kompilatorn. amxmlc-kompilatorn finns i katalogen `bin` i Flex SDK. Om du vill kan du ställa in datorns sökvägsmiljö så att den omfattar bin-katalogen i Flex SDK. Om du ställer in sökvägen blir det enklare att köra verktygen på kommandoraden.

1 Öppna ett kommandoskal eller en terminal och navigera till projektmappen för AIR-programmet.

2 Ange följande kommando:

```
amxmlc HelloWorld.as
```

Om du kör `amxmlc` skapas `HelloWorld.swf`, som innehåller den kompilerade koden i programmet.

Obs! *Om programmet inte kompileras åtgärdar du syntax- eller stavfel. Fel och varningar visas i konsolfönstret som används för att köra amxmlc-kompilatorn.*

Mer information finns i ”Kompilera MXML- och ActionScript-källfiler för AIR” på sidan 152.

Testa programmet

Om du vill köra och testa programmet från kommandoraden använder du AIR Debug Launcher (ADL) för att starta programmet med hjälp av dess programbeskrivningsfil. (ADL finns i katalogen bin i AIR SDK och Flex SDK.)

❖ I kommandotolken skriver du följande kommando:

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

Du hittar mer information i ”Enhetssimulering med ADL” på sidan 98.

Skapa APK-paketfilen

När ditt program fungerar som det ska kan du använda ADT-verktyget för att paketera programmet i en APK-paketfil. En APK-paketfil är Androids eget programfilsformat, som du kan distribuera till användarna.

Alla Android-program måste signeras. Till skillnad från AIR-filer brukar Android-program signeras med ett självsignerande certifikat. Android-operativsystemet försöker inte bekräfta programutvecklarens identitet. Du kan använda ett certifikat som genererats av ADT för att signera Android-paket. Certifikat för program som skickas till Android-marknaden måste ha en giltighet på minst 25 år.

Generera ett självsignerat certifikat och nyckelpar

❖ I kommandotolken anger du följande kommando (den körbara ADT-filen finns i katalogen `bin` i Flex SDK):

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

Detta exempel innehåller det minsta antal attribut som kan anges för ett certifikat. Nyckeltypen måste vara antingen *1024-RSA* eller *2048-RSA* (läs mer i ”ADT-kommandot certificate” på sidan 171).

Skapa AIR-paketet

❖ I kommandotolken skriver du följande kommando (på en rad):

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

Du kommer att få en fråga om nyckelfilens lösenord. Skriv lösenordet och tryck på Retur.

Mer information finns i ”Paketera AIR-mobilprogram” på sidan 91.

Installera AIR-miljön

Du kan installera den senaste versionen av AIR-miljön på enheten från Android Market. Du kan också installera den miljö som ingår i SDK på en enhet eller en Android-emulator.

❖ I kommandotolken skriver du följande kommando (på en rad):

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

Ange flaggan `-platformsdk` till katalogen med Android SDK (ange den överordnade mappen till mappen tools).

ADT installerar Runtime.apk som ingår i SDK.

Du hittar mer information i ”Installera AIR-miljön och AIR-program för utveckling” på sidan 106.

Installera AIR-programmet

❖ I kommandotolken skriver du följande kommando (på en rad):

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

Ange flaggan `-platformsdk` till katalogen med Android SDK (ange den överordnade mappen till mappen tools).

Du hittar mer information i ”Installera AIR-miljön och AIR-program för utveckling” på sidan 106.

Du kan starta ditt program genom att knacka på programikonen på enhetens eller emulatorns skärm.

Kapitel 6: Utveckla AIR-program för skrivbordet

Arbetsflöde för att utveckla ett AIR-skrivbordsprogram

Det grundläggande arbetsflödet för att utveckla AIR-program är samma som i de flesta traditionella utvecklingsmodeller: koda, kompilera, testa och, mot slutet av processen, paketera det hela i ett installationsprogram.

Du kan skriva programkoden med Flash, Flex och ActionScript och kompilera med Flash Professional, Flash Builder eller kommandoradskompilerarna mxmlc och compc. Du kan också skriva programkoden med HTML och JavaScript och hoppa över kompileringssteget.

Du kan testa AIR-skrivbordsprogram med ADL-verktyget, som kör programmet utan att det först behöver paketeras och installeras. Flash Professional, Flash Builder, Dreamweaver och Aptana kan alla integreras med Flash-felsökaren. Du kan också starta felsökningsverktyget (FDB) manuellt när du kör ADL från kommandoraden. ADL visar fel och utdata från trace-programsatser.

Alla AIR-program måste paketeras i en installationsfil. AIR-filformatet för flera plattformar rekommenderas såvida inte:

- Du måste ha åtkomst till plattformsberoende API:er såsom klassen NativeProcess.
- ANE-tillägg används i ditt program.

I så fall kan du paketera AIR-programmet som en plattformsberoende systeminstallationsfil.

SWF-baserade program

1. Skriv MXML- eller ActionScript-koden.
2. Skapa de resurser som behövs, t.ex. bitmappsfiler för ikoner.
3. Skapa programbeskrivningen.
4. Kompilera ActionScript-koden.
5. Testa programmet.
6. Paketera och signera som en AIR-fil som använder målet *air*.

HTML-baserade program

1. Skriv HTML- och ActionScript-koden.
2. Skapa de resurser som behövs, t.ex. bitmappsfiler för ikoner.
3. Skapa programbeskrivningen.
4. Testa programmet.
5. Paketera och signera som en AIR-fil som använder målet *air*.

Skapa systemspecifika installationsprogram för AIR-program

1. Skriv koden (ActionScript eller HTML och JavaScript).
2. Skapa de resurser som behövs, t.ex. bitmappsfiler för ikoner.

3 Skapa programbeskrivningen, ange profilen *extendedDesktop*.

4 Kompilera all ActionScript-kod.

5 Testa programmet.

6 Paketera programmet på varje målplattform som använder målet *native*.

Obs! *Det systemspecifika installationsprogrammet för en målplattform måste skapas på den plattformen. Du kan till exempel inte skapa ett installationsprogram för Windows på en Mac. Du kan använda en virtuell motor som exempelvis VMWare för att köra flera plattformar på samma dator.*

Skapa AIR-program med ett låst miljöpaket

1 Skriv koden (ActionScript eller HTML och JavaScript).

2 Skapa de resurser som behövs, t.ex. bitmappsfiler för ikoner.

3 Skapa programbeskrivningen, ange profilen *extendedDesktop*.

4 Kompilera all ActionScript-kod.

5 Testa programmet.

6 Paketera programmet på varje målplattform som använder målet *bundle*.

7 Skapa ett installationsprogram som använder paketfilerna. (I AIR SDK finns inte verktyg för att skapa ett sådant installationsprogram, men det finns många tredjepartsverktyg att tillgå.)

Obs! *Paketet för en målplattform måste skapas på den plattformen. Du kan till exempel inte skapa ett paket för Windows på en Mac. Du kan använda en virtuell motor som exempelvis VMWare för att köra flera plattformar på samma dator.*

Ange egenskaper för skrivbordsprogram

Ange de grundläggande programegenskaperna i programbeskrivningsfilen. I det här avsnittet behandlas de egenskaper som rör AIR-skrivbordsprogram. Elementen i programbeskrivningsfilen beskrivs ingående i ”AIR-programbeskrivningsfiler” på sidan 200.

Nödvändig version av AIR-miljön

Ange den version av AIR-miljön som krävs för ditt program med hjälp av namnutrymmet i programbeskrivningsfilen.

Namnutrymmet, som tilldelas i elementet `application`, avgör till stor del vilka funktioner ditt program kan använda. Om du i program till exempel använder AIR 1.5-namnutrymmet, och användaren har AIR 3.0 installerat, ser ditt program AIR 1.5-beteendet (även om beteendet har ändrats i AIR 3.0). Ditt program får inte tillgång till de nya beteendena och funktionerna förrän du ändrar namnutrymmet och publicerar en uppdatering. Säkerhets- och WebKit-ändringar är de huvudsakliga undantagen till den här regeln.

Ange namnutrymmet med xmlns-attributet för rotelementet `application`:

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

Fler hjälpavsnitt

”application” på sidan 205

Programidentitet

Det är flera inställningar som bör vara unika för varje program du publicerar. Bland de unika inställningarna finns ID, namn och filnamn.

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

Fler hjälpavsnitt

”id” på sidan 221

”filename” på sidan 216

”name” på sidan 229

Programversion

I tidigare versioner av AIR än AIR 2.5 anger du programversionen i elementet `version`. Du kan använda valfri sträng. AIR-miljön tolkar inte strängen, och 2.0 behandlas inte som en senare version än 1.0.

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

I AIR 2.5 och senare anger du programversionen i elementet `versionNumber`. Du kan inte längre använda elementet `version`. När du anger ett värde för `versionNumber` måste du använda en sekvens med upp till tre siffror, åtskilda med punkter, t.ex. ”0.1.2”. Varje segment i versionsnumret kan ha upp till tre siffror. (Följaktligen är ”999.999.999” det största versionsnummer som tillåts.) Du behöver inte inkludera alla tre segment i versionsnumret; "1" och "1.0" är också giltiga versionsnummer.

Du kan också ange en etikett för versionen med elementet `versionLabel`. När du lägger till en versionsetikett visas den i stället för versionsnumret i bland annat installationsdialogrutorna för AIR-programmet.

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

Fler hjälpavsnitt

”version” på sidan 237

”versionLabel” på sidan 237

”versionNumber” på sidan 238

Egenskaper för huvudfönstret

När AIR startar ett program på skrivbordet skapas ett fönster, till vilket SWF-huvudfilen eller HTML-sidan läses in. I AIR används de underordnade elementen för elementet `initialWindow` för att styra inledande utseende och beteende för det här inledande programfönstret.

- **content** – Den huvudsakliga SWF-programfilen i det underordnade `content`-elementet i `initialWindow`-elementet. När du har enheter i skrivbordsprofilen som mål kan du använda en SWF- eller HTML-fil.

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

Du måste inkludera filen i AIR-paketet (med ADT eller den integrerade utvecklingsmiljön). Det räcker inte att bara referera till namnet i programbeskrivningen för att filen automatiskt ska inkluderas i paketet.

- **depthAndStencil** – Anger att djup- eller stencilbuffert ska användas. Dessa buffertar används oftast när du arbetar med 3D-innehåll.

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

- **height** – Höjden på det inledande fönstret.
- **maximizable** – Anger om systemkontrollen för maximering av fönstret visas.
- **maxSize** – Den största tillåtna storleken.
- **minimizable** – Anger om systemkontrollen för minimering av fönstret visas.
- **minSize** – Den minsta tillåtna storleken.
- **renderMode** — I AIR 3 eller senare, kan återgivningsläget ställas in på *auto*, *cpu*, *direct* eller *gpu* för datorprogram. I tidigare versioner av AIR, ignorerades den här inställningen på datorplattformar. Inställningen renderMode kan inte ändras under körning.
  - auto – i stort sett detsamma som cpu-läge.
  - cpu – visningsobjekt återges och kopieras till visningsminnet i programmet. StageVideo är endast tillgänglig när ett fönster visas i helskärmsläge. Programvaruåtergivning används för Stage3D.
  - direct – visningsobjekt återges av körningsprogrammet, men om den återgivna bildrutan kopieras till visningsminnet (blitting) kommer den att maskinvaruacceleras. StageVideo är tillgänglig. Maskinvaruaccelereration används i Stage3D, när det är möjligt. Om fönstrets genomskinlighet är true, kommer fönstret att återgå till programvaruåtergivning och blitting.

    Obs! *För att kunna anpassa GPU-accelerationen för Flash-innehållet med AIR för mobila plattformar, rekommenderar Adobe att du använder renderMode="direct" (dvs. Stage3D) i stället för renderMode="gpu". Adobe stöder och rekommenderar följande Stage3D-baserade ramverk officiellt: Starling (2D) och Away3D (3D). Mer information om Stage3D och Starling/Away3D finns på http://gaming.adobe.com/getstarted/.*
  - gpu – maskinvaruacceleration används om det är tillgängligt.
- **requestedDisplayResolution** – Anger om programmet ska använda standardupplösning (*standard*) eller hög upplösning (*high*) på MacBook Pro-datorer med högupplösta skärmar. Värdet ignoreras på alla andra plattformar. Om värdet är *standard* återges varje pixel på scenen som fyra pixlar på skärmen. Om värdet är *high* motsvarar varje pixel på scenen en enda fysisk pixel på skärmen. Det angivna värdet används för alla programfönster. Elementet `requestedDisplayResolution` för AIR-skrivbordsprogram (som ett underordnat element för elementet `intialWindow`) finns i AIR 3.6 och senare versioner.
- **resizable** – Anger om systemkontrollen för storleksändring av fönstret visas.
- **systemChrome** – Anger om operativsystemets standard för fönsterutseende används. Det går inte att ändra inställningen systemChrome för ett fönster vid körning.
- **title** – Fönstrets titel.
- **transparent** – Anger om fönstret alfablandas mot bakgrunden. Fönstret kan inte använda systemkontroller om genomskinlighet är aktiverat. Det går inte att ändra inställningen för genomskinlighet för ett fönster vid körning.
- **visible** – Anger om fönstret är synligt så fort det skapas. Som standard är fönstret inte synligt direkt för att programmet ska hinna rita upp innehållet innan det visas.

- **width** – Fönstrets bredd.
- **x** – Fönstrets vågräta position.
- **y** – Fönstrets lodräta position.

Fler hjälpavsnitt

"content" på sidan 210

"depthAndStencil" på sidan 212

"height" på sidan 220

"maximizable" på sidan 228

"maxSize" på sidan 228

"minimizable" på sidan 229

"minimizable" på sidan 229

"minSize" på sidan 229

"renderMode" på sidan 231

"requestedDisplayResolution" på sidan 232

"resizable" på sidan 233

"systemChrome" på sidan 235

"title" på sidan 236

"transparent" på sidan 237

"visible" på sidan 238

"width" på sidan 239

"x" på sidan 239

"y" på sidan 239

Skrivbordsfunktioner

Följande element styr installations- och uppdateringsfunktioner på skrivbordet.

- customUpdateUI – Använd detta om du vill skapa egna dialogrutor för att uppdatera ett program. Om du anger det som `false` (standard) används standarddialogrutorna i AIR.
- fileTypes – Anger de filtyper som ska registreras som standard för att öppnas med ditt program. Om ett annat program redan har angetts som standard för att öppna en viss filtyp åsidosätts den befintliga registreringen inte av AIR. Ditt program kan däremot åsidosätta registreringen vid körning med metoden `setAsDefaultApplication()` för NativeApplication-objektet. Det hör till god ton att be användaren om tillåtelse att åsidosätta de befintliga filtypsassociationerna.

  Obs! *Filtypsregistrering ignoreras när du paketerar ett program i ett låst miljöpaket (med hjälp av målet* `-bundle`*). Om du vill registrera en given fil måste du skapa ett installationsprogram som utför registreringen.*
- installFolder – Anger en sökväg i förhållande till standardmappen för programinstallation i vilken programmet installeras. Du kan använda den här inställningen för att ange en anpassad mapp och för att gruppera flera program i en gemensam mapp.

- programMenuFolder – Anger menyhierarkin för Windows-menyn Alla program. Du kan använda den här inställningen för att gruppera flera program på en gemensam meny. Om ingen menymapp anges läggs programgenvägen till direkt på huvudmenyn.

Fler hjälpavsnitt

”customUpdateUI” på sidan 211

”fileTypes” på sidan 218

”installFolder” på sidan 225

”programMenuFolder” på sidan 231

Profiler som stöds

Om ditt program bara är avsett för skrivbordet kan du hindra det från att installeras på enheter i andra profiler genom att utesluta dessa profiler i listan med profiler som stöds. Om ditt program använder klassen NativeProcess eller ANE-tillägg måste du ha stöd för profilen `extendedDesktop`.

Om du utelämnar elementet `supportedProfile` från programbeskrivningen antas programmet ha stöd för alla definierade profiler. Om du vill begränsa programmet till en viss lista med profiler listar du profilerna, åtskilda med blanksteg:

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

En lista över de ActionScript-klasser som stöds i profilerna `desktop` och `extendedDesktop` finns i ”Funktioner i de olika profilerna” på sidan 242.

Fler hjälpavsnitt

”supportedProfiles” på sidan 234

Obligatoriska ANE-tillägg

Program som har stöd för profilen `extendedDesktop` kan använda ANE-tillägg.

Deklarera alla ANE-tillägg som används i AIR-programmet i programbeskrivningen. Följande exempel visar syntaxen för att ange två nödvändiga ANE-tillägg:

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

Elementet `extensionID` har samma värde som elementet `id` i tilläggsbeskrivningsfilen. Tilläggsbeskrivningsfilen är en XML-fil med namnet extension.xml. Den är paketerad i den ANE-fil som du får från ANE-utvecklaren.

Programikoner

På skrivbordet används de ikoner som angetts i programbeskrivningen som ikoner för programmets filer samt för dess genväg och programmeny. Programikoner bör anges som en uppsättning PNG-bilder med 16 x 16, 32 x 32, 48 x 48 och 128 x 128 pixlar. Ange sökvägen till ikonfilerna i icon-elementet i programbeskrivningsfilen:

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

Om du inte anger någon ikon med en bestämd storlek används den näststörsta storleken, som skalas för att passa. Om du inte anger några ikoner alls används systemets standardikon.

Fler hjälpavsnitt

”icon” på sidan 221

”imageNxN” på sidan 222

Ignorerade inställningar

Program på stationära datorer ignorerar programinställningar som gäller funktioner i mobilprofiler. De inställningar som ignoreras är:

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

Felsöka ett AIR-skrivbordsprogram

Om du utvecklar ditt program med en integrerad utvecklingsmiljö, som Flash Builder, Flash Professional eller Dreamweaver, är felsökningsverktygen vanligtvis inbyggda. Du kan felsöka programmet genom att helt enkelt starta det i felsökningsläge. Om du inte använder en integrerad utvecklingsmiljö med stöd för direkt felsökning kan du använda ADL (AIR Debug Launcher) och Flash Debugger (FDB) för att underlätta felsökningen av programmet.

Fler hjälpavsnitt

De Monsters: Monster Debugger

”Felsökning med AIR HTML Introspector” på sidan 277

Köra ett program med ADL

Du kan köra ett AIR-program utan att paketera och installera det med ADL. Skicka programbeskrivningsfilen till ADL som en parameter, enligt följande exempel (ActionScript-koden i programmet måste kompileras först):

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

ADL skriver ut trace-programsatser, körningsfel och HTML-tolkningsfel till terminalfönstret. Om en FDB-process väntar på en inkommande anslutning ansluter ADL till felsökaren.

Du kan även använda ADL för att felsöka ett AIR-program som använder ANE-tillägg. Till exempel:

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

Fler hjälpavsnitt

"AIR Debug Launcher (ADL)" på sidan 156

Skriva ut trace-programsatser

Om du vill skriva ut trace-programsatser till konsolen som används för att köra ADL, lägger du till programsatsen i koden med funktionen `trace()`.

Obs! *Om `trace()`-programsatserna inte visas på konsolen kontrollerar du att du inte har angett `ErrorReportingEnable` eller `TraceOutputFileEnable` i filen mm.cfg. Du hittar mer information om var den här filen finns på olika plattformar i Redigera filen mm.cfg.*

ActionScript-exempel:

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

JavaScript-exempel:

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

I JavaScript-kod kan du använda funktionerna `alert()` och `confirm()` om du vill visa felsökningsmeddelanden från programmet. Dessutom skrivs radnumren för syntaxfel och alla ej infångade JavaScript-undantag till konsolen.

Obs! *Du måste importera filen AIRAliases.js till sidan om du vill använda air-prefixet som visas i JavaScript-exemplet. Filen finns i katalogen frameworks i AIR SDK.*

Ansluta till Flash Debugger (FDB)

Om du vill felsöka AIR-program med Flash Debugger startar du först en FDB-session och sedan programmet med hjälp av ADL.

Obs! *I SWF-baserade AIR-program måste ActionScript-källfilerna kompileras med flaggan `-debug`. (I Flash Professional ska du markera alternativet Tillåt felsökning i dialogrutan Publiceringsinställningar.)*

1. Starta FDB. FDB-programmet finns i katalogen `bin` i Flex SDK-mappen.

   I konsolen visas FDB-kommandotolken: `<fdb>`

2. Kör kommandot `run`: `<fdb>run [Retur]`
3. Starta en annan felsökningsversion av programmet i en annan kommando- eller skalkonsol:

   ```
   adl myApp.xml
   ```

4. Ange eventuellt brytpunkter med hjälp av FDB-kommandona.
5. Typ: `continue [Retur]`

Om ett AIR-program är SWF-baserat kontrollerar felsökaren bara exekveringen av ActionScript-kod. Om ett AIR-program är HTML-baserat kontrollerar felsökaren bara exekveringen av JavaScript-kod.

Om du vill köra ADL utan att ansluta till felsökaren inkluderar du alternativet `-nodebug`:

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

Om du vill visa grundläggande information om FDB-kommandon kör du kommandot `help`:

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

Information om FDB-kommandona finns i avsnittet Using the command-line debugger commands i Flex-dokumentationen.

Paketera en AIR-installationsfil för skrivbordet

Alla AIR-program måste åtminstone ha en programbeskrivningsfil och en huvud-SWF- eller HTML-fil. Eventuella andra resurser som ska installeras med programmet måste också ingå i AIR-filen.

I den här texten beskrivs hur du paketerar ett AIR-program med de kommandoradsverktyg som ingår i SDK. Information om hur du paketerar ett program med något av Adobes redigeringsverktyg finns i följande avsnitt:

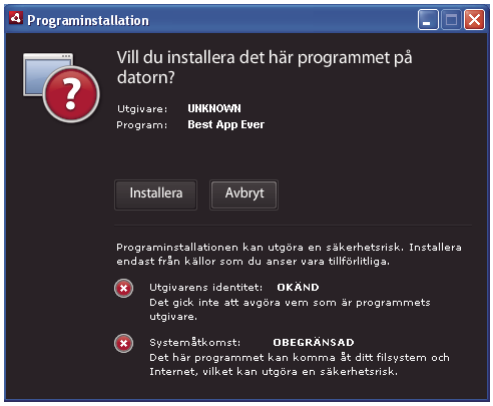
- Adobe® Flex® Builder™, se även Packaging AIR applications with Flex Builder.
- Adobe® Flash® Builder™, se även Packaging AIR applications with Flash Builder.
- Adobe® Flash® Professional, se Publicera för Adobe AIR.
- Adobe® Dreamweaver®, se Skapa ett AIR-program i Dreamweaver.

Alla AIR-installationsfiler måste signeras med ett digitalt certifikat. AIR-intallationsprogrammet använder signaturen för att kontrollera att programfilen inte har installerats sedan du signerat den. Du kan använda kodsigneringscertifikat från en certifikatutfärdare eller ett självsignerat certifikat.

När du använder ett certifikat som har utfärdats av en betrodd certifikatutfärdare bevisar du din identitet för användaren. Installationsdialogrutan visar att din identitet verifierats av certifikatutfärdaren:

Bekräftande installationsdialogruta för program som signeras av ett betrott certifikat

När du använder ett självsignerat certifikat kan användarna inte identifiera dig som utfärdaren. Ett självsignerat certifikat tyder även på att det är mer osäkert om paketet har ändrats. (På grund av att den rätta installationsfilen kan ersättas med en förfalskad fil innan den når mottagaren.) Installationsdialogrutan visar att utgivarens identitet inte kan verifieras. Användaren tar en större risk när ditt program installeras:



Bekräftande installationsdialogruta för program som signeras av ett självsignerat certifikat

Du kan paketera och signera en AIR-fil i ett enda steg med ADT-kommandot `-package`. Du kan också skapa ett mellanliggande, osignerat paket med kommandot `-prepare` och signera det mellanliggande paketet med kommandot `-sign` i ett separat steg.

Obs! *I Java-versionerna 1.5 och senare går det inte att använda hög-ASCII-tecken i lösenord för att skydda PKCS12-certifikatfiler. Använd bara vanliga ASCII-tecken i lösenordet när du skapar eller exporterar en kodsignerande certifikatfil.*

När du signerar installationspaketet kontaktar ADT automatiskt en tidstämpelbehörighetsserver för att kontrollera tiden. Tidstämpelinformationen tas med i AIR-filen. En AIR-fil som innehåller en kontrollerad tidstämpel kan installeras när som helst i framtiden. Om ADT inte kan ansluta till tidstämpelservern avbryts paketeringen. Du kan åsidosätta tidstämpelalternativet, men utan tidstämpel kan ett AIR-program inte installeras efter att certifikatet som användes för att signera installationsfilen har upphört att gälla.

När du skapar ett paket för att uppdatera ett befintligt AIR-program måste paketet signeras med samma certifikat som det ursprungliga programmet. Om det ursprungliga certifikatet har förnyats eller gått ut inom de senaste 180 dagarna, eller om du vill byta till ett nytt certifikat, kan du använda en flyttningssignatur. Med en flyttningssignatur signerar du AIR-programfilen med både det nya och det gamla certifikatet. Använd kommandot `-migrate` för att använda flyttningssignaturen, enligt beskrivningen i ”ADT-kommandot migrate” på sidan 170.

Viktigt! *Det finns en strikt frist på 180 dagar när en flyttningssignatur används efter det att det ursprungliga certifikatet har gått ut. Utan en flyttningssignatur måste befintliga användare avinstallera programmet innan de installerar den nya versionen. Fristen gäller endast för program där AIR-version 1.5.3 eller senare anges i namnutrymmet för programbeskrivningen. Det finns ingen frist för tidigare versioner av AIR körtidsmodulen.*

Flyttningssignaturer stöds inte i versioner före AIR 1.1. Du måste paketera ett program med en SDK som har version 1.1 eller senare om du vill använda en flyttningssignatur.

Program som används med AIR-filer kallas för skrivbordsprofilprogram. Du kan inte använda ADT för att paketera ett internt installationsprogram för ett AIR-program om programbeskrivningsfilen inte har stöd för skrivbordsprofilen. Du kan begränsa profilen med elementet `supportedProfiles` i programbeskrivningsfilen. Läs mer i ”Enhetsprofiler” på sidan 241 och ”supportedProfiles” på sidan 234.

Obs! *Inställningarna i programbeskrivningsfilen avgör identiteten på ett AIR-program och dess standardinstallationssökväg. Läs mer i ”AIR-programbeskrivningsfiler” på sidan 200.*

Utgivar-ID

Från och med AIR 1.5.3 har utgivar-ID tagits bort. Nya program (ursprungligen utgivna med AIR 1.5.3 eller senare) behöver inte och ska inte ange ett utgivar-ID.

När du uppdaterar program som givits ut med tidigare versioner av AIR, måste du ange det ursprungliga utgivar-ID:et i programbeskrivningsfilen. Annars behandlas den installerade versionen och den uppdaterade versionen av programmet som olika program. Om du använder ett annat ID eller utelämnar taggen publisherID, måste användaren avinstallera den föregående versionen innan den nya installeras.

Du hittar ursprungligt utgivar-ID i filen `publisherid` i undermappen META-INF/AIR där det ursprungliga programmet är installerat. Strängen i den filen är utgivar-ID:t. Programbeskrivningen måste ange runtime-versionen av AIR 1.5.3 (eller senare) i namnutrymmesdeklarationen i programbeskrivningsfilen för att ange utgivar-ID manuellt.

För program som givits ut före AIR 1.5.3 eller som givits ut med AIR 1.5.3 SDK beräknas ett utgivar-ID baserat på det signerande certifikatet när du anger en tidigare version av AIR i programbeskrivningsfilens namnutrymme. Detta ID används tillsammans med programmets ID för att avgöra programmets identitet. Eventuellt utgivar-ID används i följande syften:

- Verifiera att en AIR-fil är en uppdatering i stället för ett nytt program som ska installeras
- Som en del av krypteringsnyckeln för den krypterade lokala lagringsplatsen
- Som en del av sökvägen till programlagringskatalogen
- Som en del av anslutningssträngen för lokala anslutningar
- Som en del av identitetssträngen som används för att anropa ett program med webbläsar-API:n för AIR
- Som en del av OSID (används när man skapar anpassade installations-/avinstallationsprogram)

Före AIR 1.5.3 kunde utgivar-ID för ett program ändras om du signerade en programuppdatering med en flyttningssignatur i stället för att använda ett nytt eller förnyat certifikat. När ett utgivar-ID ändras, ändras även beteendet för eventuella AIR-funktioner som är beroende av ID:t. Det går till exempel inte längre att komma åt data på den befintliga krypterade lokala lagringsplatsen och eventuella Flash- eller AIR-instanser som skapar en lokal anslutning till programmet måste använda det nya ID:t i anslutningssträngen.

I AIR 1.5.3 och senare baseras inte utgivar-ID på det signerande certifikatet och det tilldelas bara om taggen publisherID inkluderas i programbeskrivningsfilen. Ett program kan inte uppdateras om det utgivar-ID som angetts för AIR-uppdateringspaketet inte motsvarar programmets utgivar-ID.

Paketera med ADT

Du kan använda kommandoradsverktyget ADT i AIR för att paketera ett AIR-program. Innan du paketerar programmet måste du kompilera all ActionScript-kod, MXML-kod och tilläggskod. Du måste också ha ett kodsigneringscertifikat.

Du hittar detaljerad referensinformation om ADT-kommandon och -alternativ i ”AIR Developer Tool (ADT)” på sidan 162.

Skapa ett AIR-paket

Om du vill skapa ett AIR-paket använder du ADT-kommandot package och anger måltypen som *air* för officiella versioner.

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

Exemplet förutsätter att sökvägen till ADT-verktyget finns i kommandoradens sökvägsdefinition. (Läs ”Systemvariabeln path” på sidan 299 om du behöver mer hjälp.)

Du måste köra kommandot från den katalog som innehåller programfilerna. Programfilerna i exemplet är myApp-app.xml (programbeskrivningsfilen), myApp.swf och en ikonkatalog.

När du kör kommandot så som visas tillfrågas du om lösenordet för nyckelbehållaren i ADT. (De lösenordstecken du skriver visas inte alltid. Tryck på Retur när du är klar.)

Skapa ett AIR-paket från en AIRI-fil

Du kan signera en AIRI-fil för att skapa ett installationsbart AIR-paket:

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

Paketera ett systemspecifikt installationsprogram för skrivbordet

Från och med AIR 2 kan du använda ADT för att skapa systemspecifika installationsprogram för distribution av AIR-program. Du kan t.ex. skapa en EXE-installationsfil för distribution av ett AIR-program för Windows. Du kan skapa en DMG-installationsfil för distribution av ett AIR-program för Mac OS. I AIR 2.5 and AIR 2.6 kan du skapa en DEB- eller RPM-installationsfil för distribution av ett AIR-program för Linux.

Program som installeras med ett internt installationsprogram kallas för program med utökade skrivbordsprofiler. Du kan inte använda ADT för att paketera ett internt installationsprogram för ett AIR-program om programbeskrivningsfilen inte har stöd för den utökade skrivbordsprofilen. Du kan begränsa profilen med elementet `supportedProfiles` i programbeskrivningsfilen. Läs mer i ”Enhetsprofiler” på sidan 241 och ”supportedProfiles” på sidan 234.

Du kan skapa ett internt installationsprogram för AIR-programmet på två sätt:

- Du kan skapa det interna installationsprogrammet baserat på programbeskrivningsfilen och andra källfiler. (Andra källfiler kan inkludera SWF-filer, HTML-filer och andra resurser.)
- Du kan skapa det interna installationsprogrammet baserat på en AIR-fil eller en AIRI-fil.

Du måste använda ADT i samma operativsystem som det för den interna installationsfilen som du vill generera. Om du vill skapa en EXE-fil för Windows ska du köra ADT i Windows. Om du vill skapa en DMG-fil för Mac OS ska du köra ADT i Mac OS. Om du vill skapa en DEB- eller RPG-fil för Linux ska du köra ADT från AIR 2.6 SDK i Linux.

När du skapar ett internt installationsprogram för att distribuera ett AIR-program får programmet följande funktioner:

- Det kan starta och samverka med interna processer med klassen NativeProcess. Se följande för mer information:
  - Kommunicera med interna processer i AIR (för ActionScript-utvecklare)
  - Communicating with native processes in AIR (för HTML-utvecklare)
- Det kan använda ANE-tillägg.
- Det kan använda metoden `File.openWithDefaultApplication()` för att öppna filer med det standardprogram som definierats i systemet, oavsett filtyp. (Det finns begränsningar för program som *inte* installeras med ett internt installationsprogram. Mer information finns under `File.openWithDefaultApplication()` i språkreferensen.)

När det paketeras som ett systemspecifikt installationsprogram förlorar det dock några av fördelarna med AIR-filformatet. En enda fil kan inte längre distribueras till alla stationära datorer. Den inbyggda uppdateringsfunktionen (liksom uppdateringsramverket) fungerar inte.

AIR-programmet installeras när användaren dubbelklickar på den interna installationsfilen. Om den nödvändiga versionen av Adobe AIR inte är installerad på maskinen, hämtar installationsprogrammet den från nätverket och installerar den först. Installationen misslyckas om det inte finns en nätverksanslutning med vilken rätt version av Adobe AIR kan hämtas (om det behövs). Installationen misslyckas även om Adobe AIR 2 saknar stöd för operativsystemet.

Obs! *Om du vill att en fil ska kunna köras i det installerade programmet bör du kontrollera att den är körbar i filsystemet när du paketerar programmet. (På Mac och Linux kan du använda chmod för att ange den körbara flaggan, om detta behövs.)*

Skapa ett internt installationsprogram med programmets källfiler

För att skapa ett internt installationsprogram med hjälp av programmets källfiler ska du använda `-package`-kommandot med följande syntax (på en kommandorad):

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

Syntaxen liknar den för paketering av en AIR-fil (utan ett internt installationsprogram). Det finns dock några skillnader:

- Du kan lägga till alternativet `-target native` i kommandot. (Om du anger `-target air` genererar ADT en AIR-fil i stället för en intern installationsfil.)
- Du kan ange DMG- eller EXE-målfilen som `installer_file`.
- I Windows kan du alternativt lägga till ytterligare en uppsättning signeringsalternativ som `[WINDOWS_INSTALLER_SIGNING_OPTIONS]` i syntaxen. I Windows kan du även signera Windows installationsfilen i tillägg till AIR-filen. Använd samma typ av certifikat och syntax för signeringsalternativ som du gör för att signera AIR-filen (se ”ADT-kodsigneringsalternativ” på sidan 176). Du kan använda samma certifikat för att signera AIR-filen och installationsfilen eller ange olika certifikat. När en användare hämtar en signerad Windows installationsfil från nätet, identifierar Windows källan till filen baserat på certifikatet.

Information om andra ADT-alternativ än `-target` finns i ”AIR Developer Tool (ADT)” på sidan 162.

Exemplet nedan skapar en DMG-fil (en intern installationsfil för Mac OS):

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

Exemplet nedan skapar en EXE-fil (en intern installationsfil för Windows):

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

Exemplet nedan skapar en EXE-fil och signerar den:

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

Skapa ett systemspecifikt installationsprogram för ett program där ANE-tillägg används.
Du kan skapa ett internt installationsprogram med hjälp av källfilerna för programmet och det systemspecifika tilläggspaketet som krävs för programmet. Använd kommandot `-package` med följande syntax (på en kommandorad):

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

Detta är samma syntax som används för paketering av ett systemspecifikt installationsprogram, men med ytterligare två alternativ. Använd alternativet `-extdir` *`extension-directory`* för att ange katalogen som innehåller ANE-filerna (systemspecifika tillägg) som används i programmet. Använd den valfria flaggan `-migrate` och `MIGRATION_SIGNING_OPTIONS`-parametrarna för att signera en uppdatering av ett program med en flyttningssignatur, när det primära kodsigneringscertifikatet inte är samma certifikat som det som användes av den tidigare versionen. Du hittar mer information i ”Signera en uppdaterad version av ett AIR-program” på sidan 195.

Mer information om ADT-alternativ finns i ”AIR Developer Tool (ADT)” på sidan 162.

I följande exempel skapas en DMG-fil (en systemspecifik installationsfil för Mac OS) för ett program som använder ANE-tillägg:

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

Skapa ett internt installationsprogram av en AIR-fil eller AIRI-fil
Du kan använda ADT för att generera en intern installationsfil baserad på en AIR-fil eller AIRI-fil. Använd ADT `-package`-kommandot med följande syntax (på en kommandorad) när du vill skapa en intern installationsfil baserad på en AIR-fil:

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

Syntaxen liknar den som används för att skapa en intern installationsfil baserad på AIR-programmets källfiler. Det finns dock några skillnader:

- Du anger en AIR-fil som källa i stället för en programbeskrivningsfil och andra källfiler för AIR-programmet.
- Ange inte signeringsalternativ för AIR-filen eftersom den redan är signerad

Använd ADT `-package`-kommandot med följande syntax (på en kommandorad) när du vill skapa en intern installationsfil baserad på en *AIRI*-fil:

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

Syntaxen liknar den för att skapa en intern installationsfil baserad på en AIR-fil. Det finns dock vissa skillnader:

- Du anger en AIRI-fil som källa.
- Du anger signeringsalternativ för AIR-målprogrammet.

Exemplet nedan skapar en DMG-fil (en intern installationsfil för Mac OS) baserad på en AIR-fil:

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

Exemplet nedan skapar en EXE-fil (en intern installationsfil för Windows) baserad på en AIR-fil:

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

Exemplet nedan skapar en EXE-fil (baserad på en AIR-fil) och signerar den:

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

Exemplet nedan skapar en DMG-fil (en intern installationsfil för Mac OS) baserad på en AIRI-fil:

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

Exemplet nedan skapar en EXE-fil (en intern installationsfil för Windows) baserad på en AIRI-fil:

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

I exemplet nedan skapas en EXE-fil (baserat på en AIRI-fil) och filen signeras med både en AIR-signatur och en systemspecifik Windows-signatur:

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

Paketera ett låst miljöpaket för stationära datorer

Ett låst miljöpaket är ett paket som innehåller programkod samt en speciell version av miljön. Den här typen av programpaket använder paketerade miljöer i stället för delade miljöer, som används på andra ställen i användarens dator.

Paketet som skapas är i Windows en mapp med programfiler och i Mac OS ett .app-paket. Du måste skapa paketet för ett speciellt operativsystem medan du kör operativsystemet. (En virtuell motor som exempelvis VMWare, kan användas för att köra flera operativsystem på en dator.)

Programmet kan köras från den mappen eller paketet utan att först installeras.

Fördelar

- Skapar ett självständigt program
- Ingen internetåtkomst krävs för installationen
- Programmet är isolerat från uppdateringskörningar
- Företag kan certifiera specifika program- och miljökombinationer
- Stöd för den traditionella programdistributionsmodellen
- Ingen speciell nydistribution av miljön krävs
- NativeProcess-API kan används
- ANE-tillägg kan användas
- Funktionen `File.openWithDefaultApplication()` kan användas utan restriktioner
- Kan köras från ett USB-minne eller en optisk skiva utan installation

Nackdelar

- Allvarliga säkerhetskorrigeringar är inte automatiskt tillgängliga för användare när Adobe publicerar en säkerhetsuppdatering
- Filformatet .air kan inte användas
- Du måste kanske skapa ett eget installationsprogram
- AIR-uppdaterings-API och ramverk stöds inte
- Webbläsar-API:n för AIR som används för att installera och starta ett AIR-program från en webbsida stöds inte
- I Windows måste filregistreringen hanteras i installationsprogrammet
- Kräver större utrymme för program på disken

Skapa ett låst miljöpaket för Windows

Om du vill skapa ett låst miljöpaket för Windows, måste du paketera programmet medan du kör operativsystemet Windows. Paketera programmen med ADT-målet *bundle*:

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

Med detta kommando skapas paketet i en katalog med namnet myApp. Katalogen innehåller filerna för programmet samt miljöfilerna. Du kan köra programmet direkt från mappen. Om du emellertid vill skapa en post för en programmeny, registrera filtyper eller URI-schemahanterare, måste du skapa ett installationsprogram som ställer in de begärda registerposterna. I AIR SDK inkluderas inte verktyg för att skapa sådana installationsprogram, men det finns fler alternativ från tredjpartsleverantörer, både kommersiella och gratis installationsverktyg med öppen källkod.

Genom att ange ytterligare en uppsättning av signeringsalternativ efter posten `-target bundle` på kommandoraden, kan du signera den systemspecifika körbara filen i Windows. Dessa signeringsalternativ identifierar den privata nyckeln och associerade certifikat som ska användas när den systemspecifika Windows-signaturen tillämpas. (Ett AIR-kodsigneringscertifikat kan användas.) Endast den primära körbara filen signeras. Eventuella ytterligare körbara filer, som är paketerade i programmet, signeras inte med den här processen.

Filtypsassociering
Om du vill associera ditt program med offentliga eller egna filtyper i Windows, måste du i installationsprogrammet ange de korrekta registerposterna. Filtyperna ska dessutom uppges i elementet fileTypes i programbeskrivningsfilen.

Mer information om filtyper i Windows finns på MSDN Library: File Types and File Associations

Registrering av URI-hanterare
För att ditt program ska öppna ett URL med ett givet URI-schema måste installationsprogrammet ställa in de nödvändiga registerposterna.

Mer information om hur program ska registreras för att hantera ett URI-schema finns på MSDN Library: Registering an Application to a URL Protocol

Skapa ett låst miljöpaket för Mac OS X

Om du vill skapa ett låst miljöpaket för Mac OS X, måste du paketera programmet medan du kör operativsystemet Macintosh. Paketera programmen med ADT-målet *bundle*:

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

Med detta kommando skapas programpaketet myApp.app. Paketet innehåller filerna för programmet samt miljöfilerna. Du kör programmet genom att dubbelklicka på ikonen myApp.app och installerar det genom att dra det till en lämplig plats, till exempel programmappen. Du måste emellertid, för att registrera filtyper eller hanterare för URI-scheman, redigera filen för egenskapslistan i programpaketet.

För distributionen kan du skapa en .dmg.fil (Disk Image File). I Adobe AIR SDK finns inte verktyg för att skapa en dmg-fil för ett låst miljöpaket.

Filtypsassociering
Om du vill associera ditt program med offentliga eller egna filtyper på Mac OS X, måste du redigera filen info.plist i paketet för att ställa in egenskapen CFBundleDocumentTypes. Se Mac OS X Developer Library: Information Property List Key Reference, CFBundleURLTypes.

Registrering av URI-hanterare
För att programmet ska kunna öppna ett URL med ett givet URI-schema, måste du redigera filen info.plist i paketet för att ställa in egenskapen CFBundleURLTypes. Se Mac OS X Developer Library: Information Property List Key Reference, CFBundleDocumentTypes.

Distribuera AIR-paket för stationära datorer

AIR-program kan distribueras som AIR-paket, som innehåller programkoden och alla resurser. Du kan distribuera det här paketet på något av de vanliga sätten, till exempel via nedladdning, e-post eller ett fysiskt medium som CD-ROM. Användarna kan installera programmet genom att dubbelklicka på AIR-filen. Du kan använda webbläsar-API:t för AIR (ett webbaserat ActionScript-bibliotek) för att låta användarna installera ditt AIR-program (och Adobe® AIR®, vid behov) genom att klicka på en länk på en webbsida.

AIR-program kan också paketeras och distribueras som systemspecifika installationsprogram (d.v.s. som EXE-filer på Windows, DMG-filer på Mac och DEB- eller RPM-filer på Linux). Dessa installationspaket kan distribueras och installeras i enlighet med relevanta plattformskonventioner. När du distribuerar ditt program som ett sådant systempaket förlorar du en del av fördelarna med AIR-filformatet. Det innebär att det inte längre går att använda en och samma installationsfil för de flesta plattformar, och det går inte heller att använda AIR-uppdateringsramverket eller webbläsar-API:t för AIR.

Installera och köra ett AIR-program på skrivbordet

Du kan skicka en AIR-fil till mottagaren. Du kan till exempel skicka AIR-filen som en e-postbilaga eller som en länk på en webbsida.

När användaren hämtar AIR-programmet installerar användarna det genom att följa instruktionerna nedan:

1 Dubbelklicka på AIR-filen.

Adobe AIR måste vara installerat på datorn.

2 Låt standardinställningarna vara markerade i installationsfönstret och klicka sedan på Fortsätt.

I Windows gör AIR följande automatiskt:

- Installerar programmet i katalogen Program
- Skapar en genväg till programmet på skrivbordet
- Skapar en genväg på Start-menyn
- Lägger till en post för programmet i Lägg till/ta bort program på Kontrollpanelen

I Mac OS läggs programmet som standard till i katalogen Applications.

Om programmet redan är installerat får användaren möjlighet att öppna den befintliga versionen av programmet eller uppdatera till versionen i den AIR-fil som hämtas. Installationsprogrammet identifierar programmet utifrån det program-ID och utgivar-ID som finns i AIR-filen.

3 Klicka på Slutför när installationen är klar.

För att installera en uppdaterad version av ett program i Mac OS måste användaren ha tillräcklig behörighet för att installera i programkatalogen. I Windows och Linux måste en användare ha administrativ behörighet.

Ett program kan också installera en ny version via ActionScript eller JavaScript. Mer information finns i ”Uppdatera AIR-program” på sidan 253.

När AIR-programmet har installerats dubbelklickar användaren på programikonen för att starta det, som ett vanligt program på skrivbordet.

- I Windows ska användaren dubbelklicka på programmets ikon (som installeras på skrivbordet eller i en mapp) eller välja programmet på Start-menyn.
- I Linux dubbelklickar du på programmets ikon (som finns på skrivbordet eller i en mapp) eller väljer programmet på programmenyn.
- I Mac OS ska användaren dubbelklicka på programmet i mappen där det installeras. Standardinstallationskatalogen är /Applications.

Obs! *Endast AIR-program utvecklade för AIR 2.6 eller tidigare kan installeras i Linux.*

Med AIR-funktionen *sömlös installation* kan en användare installera ett AIR-program genom att klicka på en länk på en webbsida. Med AIR-funktionen *webbläsaranrop* kan en användare köra ett AIR-program genom att klicka på en länk på en webbsida. Dessa funktioner beskrivs i följande avsnitt.

Installera och köra AIR-skrivbordsprogram från en webbsida

Med webbläsar-API:t för AIR kan du installera och köra AIR-program från en webbsida. Webbläsar-API:t för AIR finns i ett SWF-bibliotek, *air.swf*, som hanteras av Adobe. AIR SDK innehåller ett exempelprogram på badge-installation, som använder det här biblioteket för att installera, uppdatera eller starta ett AIR-program (och vid behov körningsmiljön). Du kan ändra detta exempelprogram eller skapa ett eget badge-webbprogram som använder air.swf-biblioteket direkt online.

Alla AIR-program kan installeras via en badge-installation i webbläsaren. Men det är bara program med elementet `<allowBrowserInvocation>true</allowBrowserInvocation>` i programbeskrivningsfilen som kan startas av ett badge-webbprogram.

Fler hjälpavsnitt

”Webbläsar-API:er i AIR.SWF” på sidan 245

Företagsdistribution på stationära datorer

IT-administratörer kan installera Adobe AIR-körningsversionen och AIR-programmen i tyst läge (utan användaråtgärder) med standarddistributionsverktygen för skrivbordet. IT-administratörer kan göra följande:

- Tyst installera Adobe AIR-körningsversionen med verktyg som Microsoft SMS, IBM Tivoli eller något annat utvecklingsverktyg som medger tysta installationer via en startfunktion
- Tyst installera AIR-programmet med samma verktyg som används för att distribuera körningsversionen

Mer information finns i *Administratörshandboken för Adobe AIR (http://www.adobe.com/go/learn_air_admin_guide_se)*.

Installationsloggar på stationära datorer

Installationsloggar registreras när själva AIR-körtidsmodulen eller ett AIR-program installeras. Du kan studera loggfilerna för att fastställa orsaken till eventuella installations- eller uppdateringsfel som inträffar.

Loggfilerna skapas på följande platser:

- Mac: standardsystemloggen (`/private/var/log/system.log`)

  Du kan visa systemloggen på Mac genom att öppna Systemmeddelanden (finns i mappen Verktygsprogram).
- Windows XP: `C:\Dokument och inställningar\<användarnamn>\Lokala inställningar\Programdata\Adobe\AIR\logs\Install.log`
- Windows Vista, Windows 7: `C:\Användare\<användarnamn>\AppData\Local\Adobe\AIR\logs\Install.log`
- Linux: `/home/<användarnamn>/.appdata/Adobe/AIR/Logs/Install.log`

Obs! *Dessa loggfiler skapades inte i AIR-versioner före AIR 2.*

Kapitel 7: Utveckla AIR-program för mobilenheter

AIR-program för mobilenheter distribueras som systemspecifika program. De använder enhetens programformat, inte AIR-filformatet. För tillfället har AIR stöd för Android APK-paket och iOS IPA-paket. När du väl har skapat en officiell version av ditt programpaket kan du distribuera det via plattformens standardfunktioner. För Android betyder det oftast Android Market, och för iOS är det Apple App Store.

Du kan använda AIR SDK och Flash Professional, Flash Builder eller ett annat ActionScript-utvecklingsverktyg för att skapa AIR-program för mobilenheter. HTML-baserade AIR-mobilprogram stöds inte för tillfället.

Obs! *Research In Motion (RIM) Blackberry Playbook har egna SDK:er för AIR-utveckling. Du kan läsa mer om Playbook-utveckling i RIM: BlackBerry Tablet OS Development.*

Obs! *I det här dokumentet beskrivs hur du utvecklar iOS-program med för AIR 2.6 eller senare. Program skapade med AIR 2.6+ kan installeras i iPhone 3Gs-, iPhone 4- och iPad-enheter som körs med iOS 4 eller senare. Om du vill utveckla AIR-program för tidigare versioner av iOS måste du använda AIR 2 Packager for iPhone enligt beskrivningen i Skapa iPhone-program.*

Mer information om sekretess finns i Sekretesshandboken för Adobe AIR SDK.

Mer information om fullständiga systemkrav för att köra AIR-program finns på Adobe AIR system requirements.

Konfigurera utvecklingsmiljön

Utöver de normala konfigurationskraven för AIR-, Flex- och Flash-utvecklingsmiljöerna ställs det dessutom några extra krav på mobilplattformarna. (Mer information om hur du konfigurerar den grundläggande AIR-utvecklingsmiljön finns i ”Adobe Flash plattformsverktyg för AIR-utveckling” på sidan 17.)

Android-konfiguration

Vanligtvis krävs för Android i AIR 2.6+ ingen speciell inställning. Androids ADB-verktyg innehåller AIR SDK:erna (i mappen lib/android/bin). ADB-verktyget används i AIR SDK för att installera, avinstallera och köra programpaket på enheten. Du kan även använda ADB för att visa systemloggar. Om du vill skapa och köra en Android-emulator måste du hämta det separata SDK:t för Android.

Om programmet lägger till element i elementet `<manifestAdditions>` i programbeskrivningen, som den aktuella versionen av AIR inte kan tolka som giltig, måste du installera en senare version av Android SDK. Ställ in miljövariabeln för AIR_ANDROID_SDK_HOME eller kommandoradsparametern `-platformsdk` i filsökvägen för SDK:n. I AIR:s paketeringsverktyg, ADT, används denna SDK för att validera poster i elementet `<manifestAdditions>`.

I AIR 2.5 måste du hämta en separat kopia av Android SDK från Google. Du kan ange systemvariabeln AIR_ANDROID_SDK_HOME så att den hänvisar till mappen med Android SDK. Om du inte anger den här systemvariabeln måste du ange sökvägen till Android SDK i argumentet `-platformsdk` på ADT-kommandoraden.

Fler hjälpavsnitt

”ADT-systemvariabler” på sidan 184

”Systemvariabeln path” på sidan 299

iOS-konfiguration

Om du vill installera och testa ett iOS-program på en enhet och distribuera det programmet måste du gå med i Apple iOS Developer Program (en kostnadsbelagd tjänst). När du väl är med i Apple iOS Developer Program får du tillgång till iOS Provisioning Portal, där du kan hämta de objekt och filer (från Apple) som behövs för att installera ett program på en enhet för testning och efterföljande distribution. Följande objekt och filer behövs:

- Certifikat för utveckling och distribution
- Program-ID:n
- Provisioneringsfiler för utveckling och distribution

Om utformning av mobilprogram

Det sätt på vilket mobilenheter används, och deras fysiska egenskaper, ställer stora krav på omsorgsfull kodning och design. Det är till exempel viktigt att effektivisera kod så att den körs så fort som möjligt. Men även kodoptimering har gränser; smart design som är anpassad efter enhetens begränsningar kan också bidra till att den visuella presentationen av ditt program inte överbelastar systemet.

Kod

Det är förstås alltid bra om koden är så snabb som möjligt, men den långsammare processorhastigheten på många mobilenheter innebär att kodoptimeringar för mobila plattformar ger avsevärt större utväxling. Dessutom drivs mobilenheter nästan alltid med batterier. Om det går att uppnå samma resultat med mindre arbete räcker batterierna längre.

Design

Det är viktigt att ta hänsyn till faktorer som den lilla skärmstorleken, interaktionen med pekskärmen och de många olika miljöer mobilen används i så att du kan designa ditt program på bästa sätt.

Kod och design i samverkan

Om programmet använder animeringar är återgivningsoptimering mycket viktigt. Men ofta räcker det inte med bara kodoptimering. Du måste utforma de visuella aspekterna av programmet på ett sådant sätt att koden kan återge dem effektivt.

Viktiga optimeringstekniker behandlas i handboken Optimera prestanda för Flash-plattformar. De tekniker som behandlas i handboken gäller allt Flash- och AIR-innehåll, men är avgörande för att utveckla program som fungerar bra på mobilenheter.

- Paul Trani: Tips and Tricks for Mobile Flash Devlopment
- roguish: GPU Test App AIR for Mobile
- Jonathan Campos: Optimization Techniques for AIR for Android apps
- Charles Schulze: AIR 2.6 Game Development: iOS included

Programmets livscykel

När ditt program tappar fokus till ett annat program sänks bildrutefrekvensen i AIR till 4 bildrutor per sekund och grafiken upphör att återges. Under den här bildrutefrekvensen har anslutningar till direktuppspelningsnätverk och socketar en tendens att brytas. Om ditt program inte använder sådana anslutningar kan du sänka bildrutefrekvensen ännu mer.

När det är lämpligt bör du även stoppa ljuduppspelningar och ta bort avlyssnare för geolocation- och accelerometersensorerna. AIR-objektet NativeApplication skickar aktiverings- och inaktiveringshändelser. Använd de här händelserna för att hantera övergången mellan det aktiva läget och bakgrundsläget.

De flesta mobiloperativsystem avslutar bakgrundsprogram utan någon förvarning. Genom att spara programläget ofta bör ditt program kunna återställas till ett rimligt läge, vare sig det återgår till aktiv status från bakgrunden eller startas på nytt.

Informationstäthet

Den fysiska skärmstorleken på mobila enheter är mindre än en datorskärm, men pixeltätheten (pixlar per tum) är högre. En och samma teckenstorlek ger bokstäver som rent fysiskt är mindre på en mobilskärm än på en vanlig datorskärm. Du måste ofta använda en större teckenstorlek för att texten ska vara läsbar. I allmänhet kan man säga att 14 punkter är den minsta teckenstorlek som går att läsa.

Mobila enheter används ofta i rörelse och under svaga ljusförhållanden. Fundera över hur mycket information som det är realistiskt att visa på skärmen om den ska vara läsbar. Det kan vara mindre än det som visas på en datorskärm med samma pixelmått.

Tänk också på att när användaren rör skärmen döljer fingret och handen en del av skärmen. Placera interaktiva element längs skärmens sidor och nederkant, om användaren måste interagera med dem mer än genom att bara röra vid dem.

Textinmatning

Många enheter har ett virtuellt tangentbord för textinmatning. Virtuella tangentbord kan dölja delar av skärmen och kan även vara klumpiga att använda. Undvik att förlita dig på tangentbordshändelser (med undantag för valknappar).

Den kan även vara en bra idé att implementera alternativ till inmatning i textfält. Om du till exempel vill att användaren ska ange ett numeriskt värde behöver du inte använda ett textfält. Du kan istället använda två knappar för att höja eller sänka ett värde.

Valknappar

Mobilenheter har ett antal valknappar. Valknappar är knappar som kan programmeras med olika funktioner. Följ plattformskonventionerna för de här knapparna i ditt program.

Ändringar av skärmorientering

Mobilinnehåll kan visas med stående eller liggande orientering. Fundera över hur du vill att ditt program ska hantera ändringar av skärmorienteringen. Mer information finns i Scenorientering.

Skärmnedtoning

I AIR hindras skärmen inte automatiskt från att tonas ned vid uppspelning av video. Du kan använda egenskapen `systemIdleMode` för AIR-objektet NativeApplication för att styra om enheten ska gå in i energisparläge. (På vissa plattformar måste du begära nödvändig behörighet för att den här funktionen ska fungera.)

Inkommande telefonsamtal

I AIR-miljön stängs ljudet automatiskt av när användaren ringer eller tar emot ett telefonsamtal. På Android bör du ange Android-behörigheten READ_PHONE_STATE i programbeskrivningen om ditt program spelar upp ljud när det är i bakgrunden. Annars kan AIR-miljön inte identifiera telefonsamtal och stänga av ljudet automatiskt. Läs mer i ”Android-behörigheter” på sidan 74.

Träffytor

Tänk på storleken på träffytorna när du utformar knappar och andra element i användargränssnittet som användaren kan trycka på. Gör dessa element tillräckligt stora för att lätt kunna aktiveras med ett finger på pekskärmen. Se också till att du har tillräckligt med utrymme mellan träffytorna. Träffytan bör vara ungefär 44 till 57 pixlar på var sida för en vanlig telefonskärm med hög upplösning.

Installationsstorlek för programpaket

Mobilenheter har vanligtvis betydligt mindre lagringsutrymme för installation av program och data än stationära datorer. Minimera paketstorleken genom att ta bort resurser och bibliotek som inte används.

På Android packas programpaketet inte upp till separata filer när programmet installeras. I stället expanderas resurserna till en tillfällig lagringsplats när de behövs. Du kan minimera det utrymme som krävs för den här expanderade resurslagringen genom att stänga fil- och URL-strömmar när resurserna har lästs in helt.

Filsystemåtkomst

Olika mobiloperativsystem har olika filsystembegränsningar, och dessa begränsningar brukar skilja sig från de som gäller för operativsystem på stationära datorer. Var det är bäst att spara filer och data kan därför också variera från en plattform till en annan.

En följd av variationerna i filsystemen är att genvägar till vanliga kataloger, som finns i AIR-klassen File, inte alltid är tillgängliga. Följande tabell visar vilka genvägar som kan användas på Android och iOS:

	Android	**iOS**
File.applicationDirectory	Skrivskyddad via URL (ingen systemsökväg)	Skrivskyddad
File.applicationStorageDirectory	Tillgängligt	Tillgängligt
File.cacheDirectory	Tillgängligt	Tillgängligt
File.desktopDirectory	Roten för sdcard	Inte tillgängligt
File.documentsDirectory	Roten för sdcard	Tillgängligt
File.userDirectory	Roten för sdcard	Inte tillgängligt
File.createTempDirectory()	Tillgängligt	Tillgängligt
File.createTempFile()	Tillgängligt	Tillgängligt

Apples riktlinjer för iOS-program innehåller särskilda regler om vilka lagringsplatser som ska användas för filer i olika situationer. En av dessa riktlinjer anger till exempel att endast filer som innehåller användarangivna data, eller data som inte kan återskapas eller hämtas igen, bör lagras i en katalog som är avsedd för fjärrsäkerhetskopiering. Information om hur du uppfyller Apples riktlinjer för säkerhetskopiering och cachning finns i Hantera säkerhetskopiering och cachning av filer.

Gränssnittskomponenter

Adobe har utvecklat en mobiloptimerad version av Flex-ramverket. Mer information finns i Utveckla mobilprogram med Flex och Flash Builder.

Det finns även komponentprojekt på nätet som lämpar sig för mobilprogram. Dessa inkluderar:

- Josh Tynjalas Feathers-gränssnittskontroller för Starling
- Derrick Griggs skalbara version av Minimal Comps
- Todd Andersons as3flobile-komponenter

3D-accelererad grafikåtergivning för scenen

Från och med AIR 3.2 har AIR för mobilen stöd för 3D-accelererad grafikåtergivning för scenen. ActionScript API:erna för Stage3D är en uppsättning av GPU-accelererade API:er på lågnivå för aktivering av avancerade 2D- och 3D-funktioner. Dessa lågnivå-API:er ger utvecklarna flexibilitet att anpassa GPU-maskinvaruacceleration för att få avsevärda prestandavinster. Du kan även använda spelmotorer som har stöd för ActionScript-API:er för Stage3D.

Mer information finns på Gaming engines, 3D, and Stage 3D.

Videoutjämning

Videoutjämning är inaktiverat i AIR för att ge bättre prestanda.

Systemspecifika funktioner

AIR 3.0+

Många mobilplattformar har funktioner som ännu inte är tillgängliga via standard-API:erna i AIR. Från och med AIR 3 kan du utöka AIR med egna systemspecifika kodbibliotek. Via dessa systemspecifika bibliotek kan du komma åt funktioner i operativsystemet eller funktioner som är specifika för en viss enhet. Du kan skriva ANE-tillägg i C på iOS och i Java eller C på Android. Mer information om hur du utvecklar ANE-tillägg finns i Introducing native extensions for Adobe AIR.

Arbetsflöde för att skapa AIR-program för mobilenheter

Arbetsflödet för att skapa AIR-program för mobilenheter (eller andra enheter) är i stort sett likadant som när du skapar skrivbordsprogram. De huvudsakliga skillnaderna i arbetsflödet är vid paketering, felsökning och installation av program. AIR for Android-program använder till exempel det Android-specifika APK-paketformatet i stället för AIR-paketformatet. Följaktligen använder de också Androids standardfunktioner för installation och uppdatering.

AIR for Android

När du utvecklar ett AIR-program för Android utför du oftast följande steg:

- Skriv MXML- eller ActionScript-koden.
- Skapa en AIR-programbeskrivningsfil (med 2.5-namnutrymmet eller ett senare namnutrymme).
- Kompilera programmet.
- Paketera programmet som ett Android-paket (.apk).
- Installera AIR-miljön på enheten eller Android-emulatorn (om du har en extern miljö, i den låsta miljön gäller som standard AIR 3.7 eller senare).
- Installera programmet på enheten (eller Android-emulatorn).
- Starta programmet på enheten.

Du kan använda Adobe Flash Builder, Adobe Flash Professional CS5 eller kommandoradsverktygen för att utföra de här stegen.

När ditt program är klart och paketerat som en APK-fil kan du skicka det till Android Market eller distribuera det på något annat sätt.

AIR for iOS

När du utvecklar ett AIR-program för iOS utför du oftast följande steg:

- Installera iTunes.
- Generera nödvändiga utvecklarfiler och ID:n på Apple iOS Provisioning Portal. Bland dessa finns:
  - Utvecklarcertifikat
  - Program-ID
  - Provisioneringsprofil

  Du måste lista ID:n för alla testenheter på vilka du tänker installera programmet när du skapar provisioneringsprofilen.
- Konvertera utvecklingscertifikatet och den privata nyckeln till en P12-nyckelbehållarfil.
- Skriv programmets MXML- eller ActionScript-kod.
- Kompilera programmet med en ActionScript- eller MXML-kompilerare.
- Skapa ikoner och inledande skärmbilder för programmet.
- Skapa programbeskrivningen (med 2.6-namnutrymmet eller ett senare namnutrymme).
- Paketera IPA-filen med ADT.
- Använd iTunes för att placera provisioneringsprofilen på testenheten.
- Installera och testa programmet på iOS-enheten. Du kan använda antingen iTunes eller ADT via USB (USB-stöd i AIR 3.4 och senare) för att installera IPA-filen.

När ditt AIR-program är klart kan du paketera om det med ett distributionscertifikat och en provisioneringsprofil. Sedan är det klart för att skickas till Apple App Store.

Ange egenskaper för mobilprogram

Precis som i andra AIR-program anger du de grundläggande programegenskaperna i programbeskrivningsfilen. I mobilprogram ignoreras vissa skrivbordsspecifika egenskaper, som fönsterstorlek och genomskinlighet. Mobilprogram kan också använda egna plattformsspecifika egenskaper. Du kan till exempel inkludera ett `android`-element för Android-program och ett `iPhone`-element för iOS-program.

Vanliga inställningar

Det finns flera programbeskrivningsinställningar som är viktiga för alla mobilprogram.

Nödvändig version av AIR-miljön

Ange den version av AIR-miljön som krävs för ditt program med hjälp av namnutrymmet i programbeskrivningsfilen.

Namnutrymmet, som tilldelas i elementet `application`, avgör till stor del vilka funktioner ditt program kan använda. Om ditt program till exempel använder AIR 2.7-namnutrymmet, och användaren har en senare version installerad, ser ditt program ändå AIR 2.7-beteendet (även om beteendet har ändrats i den senare versionen). Ditt program får inte tillgång till de nya beteendena och funktionerna förrän du ändrar namnutrymmet och publicerar en uppdatering. Säkerhetskorrigeringar är viktiga undantag till den här regeln.

På enheter där miljön och programmet är separata, t.ex. Android i AIR 3.6, uppmanas användaren att installera eller uppgradera AIR om de inte har den version som krävs. På enheter där program och miljö är integrerade, t.ex. iPhone, inträffar inte detta (eftersom den nödvändiga versionen paketeras med programmet från början).

Obs! *(AIR 3.7 eller senare) Som standard förpackas ADT-miljön med Android-programmen.*

Ange namnutrymmet med xmlns-attributet för rotelementet `application`. Följande namnutrymmen ska användas för mobilprogram (beroende på vilken mobilplattform du skriver för):

```
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">
```

Obs! *Stöd för iOS 3-enheter finns i Packager for iPhone SDK, som bygger på AIR 2.0 SDK. Information om hur du skapar AIR-program för iOS 3 finns i Skapa iPhone-program. I AIR 2.6 SDK (och senare) finns stöd för iOS 4, och senare, för iPhone 3Gs-, iPhone 4- och iPad-enheter.*

Fler hjälpavsnitt

”application” på sidan 205

Programidentitet

Det är flera inställningar som bör vara unika för varje program du publicerar. Bland dessa finns ID, namn och filnamn.

ID:n för Android-program

På Android konverteras ID:t till Android-paketnamnet genom att ”air.” läggs till som prefix i AIR-ID:t. Om AIR-ID:t är *com.example.MyApp* blir således Android-paketnamnet *air.com.example.MyApp*.

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

Om ID:t inte är ett giltigt paketnamn på Android konverteras det dessutom till ett giltigt namn. Bindestreckstecken ändras till understreck och inledande siffror i ID-komponenter föregås av ett versalt A. ID:t *3-goats.1-boat* ändras till exempel till paketnamnet *air.A3_goats.A1_boat*.

Obs! *Det prefix som läggs till program-ID:t kan användas för att identifiera AIR-program i Android Market. Om du inte vill att ditt program ska identifieras som ett AIR-program på grund av prefixet måste du packa upp APK-filen, ändra program-ID:t och paketera om det enligt beskrivningen i Opt-out of AIR application analytics for Android.*

ID:n för iOS-program

Ange ID:t för AIR-programmet så att det matchar det program-ID du skapade på Apple iOS Provisioning Portal.

Program-ID:n för iOS innehåller ett källpaket-ID följt av en källidentifierare. Källpaket-ID består av en teckensträng, som exempelvis 5RM86Z4DJM, som Apple tilldelar till program-ID:t. Källidentifieraren innehåller ett namn med omvänd DNS-notation som du väljer själv. Källidentifieraren kan sluta med en asterisk (*), vilket är ett jokertecken för program-ID:n. Om källidentifieraren slutar med jokertecknet kan du ersätta det med valfri giltig sträng.

Till exempel:

- Om ditt Apple program-ID är `5RM86Z4DJM.com.example.helloWorld` måste du använda `com.example.helloWorld` i programbeskrivningen.
- Om ditt Apple program-ID är `96LPVWEASL.com.example.*` (ett program-ID med jokertecken) kan du använda `com.example.helloWorld`, `com.example.anotherApp` eller något annat ID som börjar på `com.example`.
- Slutligen, om ditt Apple program-ID bara är källpaket-ID:t och ett jokertecken, t.ex. `38JE93KJL.*`, kan du använda valfritt program-ID i AIR.

När du anger program-ID:t tar du inte med delen med källpaket-ID:t i program-ID:t.

Fler hjälpavsnitt

”id” på sidan 221

”filename” på sidan 216

”name” på sidan 229

Programversion

I AIR 2.5 och senare anger du programversionen i elementet `versionNumber`. Du kan inte längre använda elementet `version`. När du anger ett värde för `versionNumber` måste du använda en sekvens med upp till tre siffror, åtskilda med punkter, t.ex. ”0.1.2”. Varje segment i versionsnumret kan ha upp till tre siffror. (Följaktligen är ”999.999.999” det största versionsnummer som tillåts.) Du behöver inte inkludera alla tre segment i versionsnumret; "1" och "1.0" är också giltiga versionsnummer.

Du kan också ange en etikett för versionen med elementet `versionLabel`. När du lägger till en versionsetikett visas den i stället för versionsnumret på bland annat programmets informationsskärm i Android. Du måste ange en versionsetikett för program som distribueras via Android Market. Om du inte anger ett `versionLabel`-värde i AIR-programbeskrivningen tilldelas `versionNumber`-värdet till Android-fältet för versionsetikett.

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

På Android översätts `versionNumber` i AIR till Android-heltalet `versionCode` enligt följande formel: `a*1000000 + b*1000 + c`, där a, b och c är komponenterna i AIR-versionsnumret: `a.b.c`.

Fler hjälpavsnitt

”version” på sidan 237

”versionLabel” på sidan 237

”versionNumber” på sidan 238

Huvudsaklig SWF-programfil

Ange den huvudsakliga SWF-programfilen i det underordnade `content`-elementet för `initialWindow`-elementet. När du har enheter i mobilprofilen som mål måste du använda en SWF-fil (HTML-baserade program stöds inte).

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

Du måste inkludera filen i AIR-paketet (med ADT eller den integrerade utvecklingsmiljön). Det räcker inte att bara referera till namnet i programbeskrivningen för att filen automatiskt ska inkluderas i paketet.

Egenskaper för huvudskärmen

Flera underordnade element till elementet initialWindow styr det inledande utseendet och beteendet för huvudprogramskärmen.

- **aspectRatio** – Anger om programmet initialt ska visas i formaten *portrait* (höjden större än bredden), *landscape* (höjden mindre än bredden) eller i *any* (scenen anpassas till alla orienteringar).

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

- **autoOrients** – Anger om scenens orientering ska ändras automatiskt när användaren roterar enheten eller utför andra åtgärder som påverkar orienteringen, som att öppna eller stänga ett utfällbart tangentbord. Om värdet är *false*, vilket är standard, ändras inte scenens orientering med enheten.

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

- **depthAndStencil** – Anger att djup- eller stencilbuffert ska användas. Dessa buffertar används oftast när du arbetar med 3D-innehåll.

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

- **fullScreen** – Anger om programmet får uppta hela enhetens skärm eller om det ska dela skärmen med vanliga systemkontroller, t.ex. ett systemstatusfält.

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

- **renderMode** – Anger om miljön ska återge programmet med grafikprocessorn (GPU) eller huvudprocessorn (CPU). I allmänhet ökar GPU-återgivning återgivningshastigheten, men en del funktioner, som vissa blandningslägen och PixelBender-filter, är inte tillgängliga i GPU-läge. Dessutom har olika enheter och olika enhetsdrivrutiner olika GPU-funktioner och begränsningar. Du bör alltid testa ditt program på så många olika enheter som möjligt, särskilt om du använder GPU-läge.

  Du kan ställa in återgivningsläget på *gpu*, *cpu*, *direct* eller *auto*. Standardvärdet är *auto*, som för tillfället har CPU-läge som reservalternativ.

  Obs! *För att kunna anpassa GPU-accelerationen för Flash-innehållet med AIR för mobila plattformar, rekommenderar Adobe att du använder renderMode="direct" (dvs. Stage3D) i stället för renderMode="gpu". Adobe stöder och rekommenderar följande Stage3D-baserade ramverk officiellt: Starling (2D) och Away3D (3D). Mer information om Stage3D och Starling/Away3D finns på http://gaming.adobe.com/getstarted/.*

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

Obs! *Du kan inte använda renderMode="direct" för program som körs i bakgrunden.*

Begränsningarna för GPU-läget är:

- I Flex-ramverket finns inget stöd för GPU-återgivningsläget.
- Filter stöds inte
- PixelBender-blandningar och fyllningar stöds inte
- Följande blandningslägen stöds inte: layer, alpha, erase, overlay, hardlight, lighten och darken
- Du bör inte använda GPU-återgivningsläget i ett program som spelar upp video.
- I GPU-återgivningsläget flyttas textfält inte på rätt sätt till synliga områden när det virtuella tangentbordet öppnas. För att garantera att textfältet syns när användaren skriver in text använder du egenskapen softKeyboardRect för scenen tillsammans med tangentbordshändelser för att flytta textfältet till det synliga området.
- Om ett visningsobjekt inte kan återges av grafikprocessorn visas det inte alls. Om till exempel ett filter används på ett visningsobjekt visas det objektet inte alls.

Obs! *GPU-implementeringen för iOS i AIR 2.6+ skiljer sig avsevärt från implementeringen i den tidigare AIR 2.0-versionen. Du behöver ta hänsyn till andra optimeringsfaktorer.*

Fler hjälpavsnitt

"aspectRatio" på sidan 208

"autoOrients" på sidan 209

"depthAndStencil" på sidan 212

"fullScreen" på sidan 220

"renderMode" på sidan 231

Profiler som stöds

Du kan lägga till elementet `supportedProfiles` för att ange vilka enhetsprofiler ditt program har stöd för. Använd profilen mobileDevice för mobilenheter. När du kör programmet med ADL (Adobe Debug Launcher) används den första profilen i listan som aktiv profil. Du kan också använda flaggan `-profile` när du kör ADL för att välja en viss profil i listan med profiler som stöds. Om ditt program fungerar med alla profiler kan du utelämna elementet `supportedProfiles` helt. I så fall används skrivbordsprofilen som aktiv standardprofil i ADL.

Om du vill ange att ditt program har stöd för både mobil- och skrivbordsprofiler, och du vanligtvis vill testa programmet i mobilprofilen, lägger du till följande element:

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

Fler hjälpavsnitt

"supportedProfiles" på sidan 234

"Enhetsprofiler" på sidan 241

"AIR Debug Launcher (ADL)" på sidan 156

Obligatoriska ANE-tillägg

Program som har stöd för profilen `mobileDevice` kan använda ANE-tillägg.

Deklarera alla ANE-tillägg som används i AIR-programmet i programbeskrivningen. Följande exempel visar syntaxen för att ange två nödvändiga ANE-tillägg:

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

Elementet `extensionID` har samma värde som elementet `id` i tilläggsbeskrivningsfilen. Tilläggsbeskrivningsfilen är en XML-fil med namnet extension.xml. Den är paketerad i den ANE-fil som du får från ANE-utvecklaren.

Beteende för virtuella tangentbord

Ange elementet `softKeyboardBehavior` som `none` om du vill inaktivera beteendet med automatiskt panorering och storleksändring, som används av miljön för att se till att det textinmatningsfält som har fokus är synligt efter att det virtuella tangentbordet har öppnats. Om du inaktiverar det automatiska beteendet måste ditt program se till att textinmatningsområdet, eller annat relevant innehåll, fortfarande är synligt efter att tangentbordet har öppnats. Du kan använda egenskapen `softKeyboardRect` för scenen tillsammans med SoftKeyboardEvent för att identifiera när tangentbordet öppnas och avgöra vilket område det täcker.

Aktivera det automatiska beteendet genom att ange elementvärdet som `pan`:

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

Eftersom `pan` är standardvärdet aktiveras det automatiska beteendet för tangentbordet också om du utelämnar elementet `softKeyboardBehavior`.

Obs! *Om du även använder GPU-återgivning stöds inte panorering.*

Fler hjälpavsnitt

”softKeyboardBehavior” på sidan 233

Stage.softKeyboardRect

SoftKeyboardEvent

Android-inställningar

På Android-plattformen kan du använda elementet `android` i programbeskrivningen för att lägga till information i Androids manifestfil, som är en programegenskapsfil som används av operativsystemet Android. Android-filen Manifest.xml genereras automatiskt i ADT när du skapar APK-paketet. AIR anger ett antal egenskaper till de värden som krävs för att vissa funktioner ska fungera. Alla andra egenskaper som anges i android-avsnittet av AIR-programbeskrivningen läggs till i motsvarade avsnitt i filen Manifest.xml.

Obs! *För de flesta AIR-program måste du ange de Android-behörigheter som krävs för ditt program i elementet* `android`*, men för det mesta behöver du inte ange några andra egenskaper.*

Du kan bara ange attribut som accepterar strängvärden, heltalsvärden eller booleska värden. Du kan inte ange referenser till resurser i programpaketet.

Obs! *För körmiljön krävs en SDK med version 14 eller senare. Om du vill skapa ett program enbart för senare versioner, ska du kontrollera att manifestet innehåller* `<uses-sdk android:minSdkVersion=""></uses-sdk>` *och tillämplig version.*

Reserverade inställningar i Androids manifestfil

AIR anger flera manifest-poster i den genererade manifestfilen för Android för att garantera att programmet och körningsfunktionerna fungerar korrekt. Du kan inte definiera följande inställningar:

manifest-element
Du kan inte ange följande attribut för manifest-elementet:

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

activity-element
Du kan inte ange följande attribut för det huvudsakliga activity-elementet:

- android:label
- android:icon

application-element
Du kan inte ange följande attribut för application-elementet:

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

Android-behörigheter

Android-säkerhetsmodellen kräver att alla program begär rätt behörighet för att använda funktioner som påverkar säkerheten eller användarens integritet. Dessa behörigheter måste anges när programmet paketeras och kan inte ändras vid körning. Via Android-operativsystemet får användaren veta vilken behörighet ett program begär när användaren installerar det. Om en behörighet som krävs för en funktion inte begärs kan Android-operativsystemet generera ett undantagsfel när ditt program försöker använda den funktionen, men det är inte säkert. Undantagsfel skickas till ditt program av miljön. Om det uppstår ett tyst fel läggs ett meddelande om behörighetsfel till i Android-systemloggen.

I AIR anger du Android-behörigheter i elementet `android` i programbeskrivningen. Följande format används för att lägga till behörigheter (där PERMISSION_NAME är namnet på en Android-behörighet):

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

Satserna uses-permissions i `manifest`-elementet läggs till direkt i Androids manifestfil.

Följande behörigheter krävs för att använda olika AIR-funktioner:

ACCESS_COARSE_LOCATION Tillåter programmet att komma åt platsdata för WIFI och mobilnätverk via klassen Geolocation.

ACCESS_FINE_LOCATION Tillåter programmet att komma åt GPS-data via klassen Geolocation.

ACCESS_NETWORK_STATE och ACCESS_WIFI_STATE Tillåter programmet att komma åt nätverksinformation via klassen NetworkInfo.

CAMERA Tillåter programmet att komma åt kameran.

Obs! *När ditt program begär behörighet att använda kamerafunktionen antar Android att programmet också behöver kameran. Om kameran är en valfri funktion i ditt program bör du lägga till ett `uses-feature`-element i manifestet för kameran och ange attributet required som `false`. Läs mer i "Kompatibilitetsfiltrering för Android" på sidan 77.*

INTERNET Tillåter programmet att utföra nätverksbegäranden och fjärrfelsökning.

READ_PHONE_STATE Tillåter att AIR-miljön stänger av ljudet under telefonsamtal. Du bör ange den här behörigheten om programmet spelar upp ljud när det är i bakgrunden.

RECORD_AUDIO Tillåter programmet att komma åt mikrofonen.

WAKE_LOCK och DISABLE_KEYGUARD Tillåter att programmet hindrar enheten från att försättas i viloläge via inställningarna för klassen SystemIdleMode.

WRITE_EXTERNAL_STORAGE Tillåter programmet att skriva till enhetens externa minneskort.

Om du till exempel vill ange behörigheter för ett program som behöver alla behörigheter kan du lägga till följande i programbeskrivningen:

```
<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>
```

Fler hjälpavsnitt

Säkerhet och behörigheter för Android (på engelska)

Android-klassen Manifest.permission

Anpassade URI-scheman för Android

Du kan använda ett anpassat URI-schema för att starta ett AIR-program från en webbsida eller ett Android-program. Stödet för anpassade URI-scheman beror på vilka intent-filter som angetts i Androids manifestfil, så den här tekniken kan inte användas på andra plattformar.

Om du vill använda en anpassad URI lägger du till ett intent-filter i programbeskrivningen inuti `<android>`-blocket. Båda `intent-filter`-elementen i följande exempel måste anges. Redigera satsen `<data android:scheme="`*`my-customuri`*`"/>` så att den speglar URI-strängen för det anpassade schemat.

```
<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>
```

Ett intent-filter talar om för Android-operativsystemet att ditt program är tillgängligt för att utföra en viss åtgärd. När det gäller en anpassad URI betyder det att användaren har klickat på en länk med det URI-schemat (och webbläsaren kan inte hantera den).

När ditt program anropas via en anpassad URI skickar NativeApplication-objektet en `invoke`-händelse. URL:en för den länken, inklusive frågeparametrar, placeras i `arguments`-arrayen för InvokeEvent-objektet. Du kan använda valfritt antal intent-filter.

Obs! *Länkar i en StageWebView-instans kan inte öppna URL:er där egna URI-scheman används.*

Fler hjälpavsnitt

Intent-filter i Android (på engelska)

Android-åtgärder och -kategorier (på engelska)

Kompatibilitetsfiltrering för Android

I Android-operativsystemet används ett antal element i programmets manifestfil för att avgöra om ditt program är kompatibelt med en viss enhet. Det är valfritt att lägga till den här informationen i manifestfilen. Om du inte inkluderar de här elementen kan ditt program installeras på alla Android-enheter. Det kan dock hända att det inte fungerar som det ska på alla Android-enheter. Ett kameraprogram är till exempel inte särskilt användbart på en telefon som saknar kamera.

De Android Manifest-taggar som du kan använda för filtrering omfattar:

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

Kameraprogram

Om du begär kamerabehörighet för ditt program antar Android att programmet behöver alla tillgängliga kamerafunktioner, inklusive autofokus och blixt. Om ditt program inte behöver alla kamerafunktioner, eller om kameran är en valfri funktion, bör du ange `uses-feature`-elementen för kameran för att tala om att dessa är valfria. Annars kommer användare vars enheter saknar en funktion, eller saknar kamera helt, inte att kunna hitta ditt program på Android Market.

Följande exempel visar hur du begär behörigheter för kameran och anger alla kamerafunktioner som valfria:

```
<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>
```

Ljudinspelningsprogram

Om du begär behörighet att spela in ljud antar Android även att programmet behöver en mikrofon. Om ljudinspelning är en valfri funktion i ditt program kan du lägga till en uses-feature-tagg för att ange att mikrofonen inte behövs. Annars kommer användare, vars enheter saknar en mikrofon, inte att kunna hitta ditt program på Android Market.

Följande exempel visar hur du begär behörighet att använda mikrofonen samtidigt som mikrofonmaskinvaran förblir valfri:

```
<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>
```

Fler hjälpavsnitt

För Android-utvecklare: Android-kompatibilitet (på engelska)

För Android-utvecklare: Konstanter för Android-funktionsnamn (på engelska)

Installationsplats

Du kan tillåta att ditt program installeras eller flyttas till ett externt minneskort genom att ange attributet `installLocation` för Androids `manifest`-elementet som antingen `auto` eller `preferExternal`:

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

Det finns dock ingen garanti för att ditt program installeras på det externa minneskortet. En användare kan också flytta programmet mellan interna och externa minneskort med programmet för systeminställningar.

Även om programmet installeras på ett extern minneskort lagras programcache och användardata (t.ex. innehållet i katalogen app-storage, delade objekt och tillfälliga filer) i det interna minnet. För att undvika att använda för mycket internt minne bör du vara selektiv när det gäller vilka data som sparas i programkatalogen. Stora mängder data bör sparas på SD-kortet med hjälp av platserna `File.userDirectory` eller `File.documentsDirectory` (som båda mappar till roten på SD-kortet på Android).

Aktivera Flash Player och andra plugin-program i ett StageWebView-objekt

I Android 3.0+ måste ett program aktivera maskinvaruaccelerationen i Android-programelementet för att plugin-innehåll ska visas i ett StageWebView-objekt. Du aktiverar plugin-återgivning genom att ange att attributet `android:hardwareAccelerated` i `application`-elementet ska vara `true`:

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

Färgdjup

AIR 3+

I AIR 3 och senare ställs visningen in för att i miljön återge 32-bitars färger. I tidigare versioner av AIR används i miljön 16-bitars färg. Genom att använda elementet <colorDepth> i programbeskrivningen kan du i miljön ange att 16-bitars färg ska användas:

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

Om du använder 16-bitars färgdjup kan återgivningsprestandan förbättras, men på bekostnad av färgåtergivningen.

Inställningar för iOS

Inställningar som bara gäller iOS-enheter placeras inuti `<iPhone>`-elementet i programbeskrivningen. `iPhone`-element kan innehålla elementen `InfoAdditions`, `requestedDisplayResolution`, `Entitlements`, `externalSwfs` och ett underordnat `forceCPURenderModeForDevices`-element.

Med `InfoAdditions`-elementet kan du ange nyckelvärdepar som läggs till i inställningsfilen Info.plist för programmet. Följande värden anger till exempel statusfältets format i programmet och att programmet inte kräver ständig trådlös nätverksåtkomst.

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

InfoAdditions-inställningarna anges i en `CDATA`-tagg.

Elementet `Entitlements` använder du för att ange nyckelvärdepar tillagda i inställningsfilen Entitlements.plist för programmet. Inställningar i Entitlements.plist ger programåtkomst till vissa iOS-funktioner, till exempel som push-meddelanden.

Mer information om Info.plist- och Entitlements.plist-inställningar finns i Apples dokumentation för utvecklare.

Stöd för bakgrundsåtgärder på iOS

AIR 3.3

Adobe AIR 3.3 och senare versioner har stöd för flera samtidiga åtgärder på iOS när vissa bakgrundsbeteenden är aktiverade:

- Ljud
- Platsuppdateringar
- Nätverk
- Välja bort programkörning i bakgrunden

Obs! *Med swf-version 21 och dess tidigare versioner har AIR inte stöd för bakgrundskörning på iOS och Android med återgivningsläget Direkt. På grund av den här begränsningen kan Stage3D-baserade program inte köra bakgrundsåtgärder som exempelvis ljuduppspelning, platsuppdateringar och nätverksöverföringar. iOS tillåter inte OpenGLES eller återgivningsanrop i bakgrunden. Program som försöker köra OpenGL-anrop i bakgrunden avslutas av iOS. Android begränsar inte program från att köra OpenGLES-anrop i bakgrunden eller från att köra andra bakgrundsåtgärder (som ljuduppspelning). Med swf-version 22 och senare kan AIR-mobilprogram köras i bakgrunden med återgivningsläget Direkt. AIR iOS-miljön genererar ett ActionScript-fel (3768 – Programmeringsgränssnittet för Stage3D kan inte användas under bakgrundskörning) om OpenGLES-anrop sker i bakgrunden. Däremot genereras inget fel på Android, eftersom Android-program tillåts köra OpenGLES-anrop i bakgrunden. För att utnyttja mobilresurserna optimalt bör återgivningsanrop inte ske när ett program körs i bakgrunden.*

Bakgrundsljud

Om du vill aktivera uppspelning och inspelning av ljud i bakgrunden inkluderar du följande nyckelvärdepar i elementet `InfoAdditions`:

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

Platsuppdateringar i bakgrunden

Om du vill aktivera platsuppdateringar i bakgrunden inkluderar du följande nyckelvärdepar i elementet `InfoAdditions`:

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

Obs! *Använd bara den här funktionen när det behövs, eftersom plats-API:er tär på batteriet.*

Bakgrundsnätverk

För att korta åtgärder ska kunna köras i bakgrunden anges egenskapen `NativeApplication.nativeApplication.executeInBackground` som `true` i programmet.

Till exempel kan en filöverföringsåtgärd startas i programmet, varefter användaren placerar ett annat program i fokus. När programmet tar emot en händelse om slutförd överföring, kan det ställa in `false` för `NativeApplication.nativeApplication.executeInBackground`.

Att egenskapen `NativeApplication.nativeApplication.executeInBackground` anges som `true` garanterar inte att programmet körs oavbrutet, eftersom det finns en tidsgräns i iOS för bakgrundsåtgärder. När iOS avslutar bakgrundsbearbetningen skickas händelsen `NativeApplication.suspend` i AIR.

Välja bort programkörning i bakgrunden

Du kan ange att programmet explicit ska välja bort körning i bakgrunden genom att inkludera följande nyckelvärdepar i elementet `InfoAdditions`:

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

Reserverade inställningar för InfoAdditions i iOS

AIR anger flera poster i den genererade filen Info.plist för att garantera att programmet och körningsfunktionerna fungerar korrekt. Du kan inte definiera följande inställningar:

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

Obs! *Du kan definiera MinimumOSVersion. Definitionen MinimumOSVersion kan användas i Air 3.3 och senare.*

Stöd för olika iOS-modeller

För iPad-stöd inkluderar du korrekt nyckelvärdesinställning för `UIDeviceFamily` inuti `InfoAdditions`-elementet. Inställningen `UIDeviceFamily` är en array med strängar. Varje sträng definierar enheter som stöds. Inställningen `<string>1</string>` definierar stöd för iPhone och iPod touch. Inställningen `<string>2</string>` definierar stöd för iPad. Inställningen `<string>3</string>` definierar stöd för tvOS. Om du bara anger en av dessa strängar stöds bara den enhetsgruppen. Följande sträng begränsar till exempel stödet till iPad:

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

Följande inställning har stöd för båda enhetsgrupperna (iPhone/iPod Touch och iPad):

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

Dessutom kan du i AIR 3.7 och senare använda taggen `forceCPURenderModeForDevices` om du vill framtvinga CPU-återgivningsläget för en angiven uppsättning enheter och aktivera GPU-återgivningsläget för återstående iOS-enheter.

Du lägger till denna tagg som underordnad till `iPhone` -taggen och skapar en blankstegsavgränsad lista över enhetsmodellnamnen. Se ”forceCPURenderModeForDevices” på sidan 219 om du vill få en lista över godkända enhetsmodellnamn.

Om du till exempel vill använda CPU-läget i gamla iPod-, iPhone- och iPad-modeller och aktiverar GPU-läget för alla andra enheter ska du ange följande i programbeskrivningen:

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

Skärmar med hög upplösning

Elementet `requestedDisplayResolution` anger om ditt program ska använda *standardupplösning* eller *hög upplösning* på iOS-enheter med högupplösta skärmar.

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

I läget med hög upplösning kan du adressera varje högupplöst pixel individuellt. I standardläget visas enhetens skärm som en skärm med standardupplösning för programmet. När en enda pixel ritas i det här läget anges färgen för fyra pixlar på den högupplösta skärmen.

Standardinställningen är `standard`. Tänk på att om målenheterna är iOS-enheter använder du elementet `requestedDisplayResolution` som ett underordnat element till elementet `iPhone` (inte elementet `InfoAdditions` eller `initialWindow`).

Om du vill använda olika inställningar för olika enheter anger du standardvärdet som värdet för elementet `requestedDisplayResolution`. Använd attributet `excludeDevices` för att ange de enheter som ska använda motsatt värde. Med följande kod används till exempel läget med hög upplösning för alla enheter som har stöd för detta, utom 3:e generationens iPad-enheter, som använder standardläget:

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

Attributet `excludeDevices` finns i AIR 3.6 och senare versioner.

Fler hjälpavsnitt

”requestedDisplayResolution” på sidan 232

Renaun Erickson: Developing for both retina and non-retina iOS screens using AIR 2.6

Anpassade URI-scheman för iOS

Du kan registrera ett anpassat URI-schema om du vill tillåta ditt program att anropas av en länk på en webbsida eller ett annat, inbyggt program på enheten. Om du vill registrera ett URI-schema lägger du till en CFBundleURLTypes-nyckel i InfoAdditions-elementet. I följande exempel registreras ett URI-schema med namnet *com.example.app* för att tillåta ett program att anropas av webbadresser med adressformatet: *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>
```

När ditt program anropas via en anpassad URI skickar NativeApplication-objektet en `invoke`-händelse. URL:en för den länken, inklusive frågeparametrar, placeras i `arguments`-arrayen för InvokeEvent-objektet. Du kan använda valfritt antal anpassade URI-scheman.

Obs! *Länkar i en StageWebView-instans kan inte öppna URL:er där egna URI-scheman används.*

Obs! *Om ett annat program redan har registrerat ett schema kan ditt program inte ersätta det som registrerat program för det URI-schemat.*

Kompatibilitetsfiltrering för iOS

Lägg till poster i en UIRequiredDeviceCapabilities-array inuti `InfoAdditions`-elementet om ditt program bara ska användas på enheter med vissa maskinvaru- eller programvarufunktioner. Följande post talar till exempel om att ett program kräver en stillbildskamera och en mikrofon:

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

Om en enhet saknar dessa funktioner kan programmet inte installeras. De funktionsinställningar som rör AIR-program är:

telefoni	kamerablixt
wifi	videokamera
sms	accelerometer
stillbildskamera	platstjänster
kamera med autofokus	gps
framåtriktad kamera	mikrofon

I AIR 2.6+ läggs *armv7* och *opengles-2* automatiskt till i listan över obligatoriska funktioner.

Obs! *Du behöver inte inkludera dessa funktioner i programbeskrivningen för att ditt program ska kunna använda dem. Använd bara UIRequiredDeviceCapabilities-inställningarna för att hindra användare från att installera ditt program på enheter som programmet inte fungerar korrekt på.*

Avsluta i stället för att pausa

När en användare växlar från ett AIR-program placeras detta i bakgrunden och pausas. Om du vill att ditt program ska avslutas helt i stället för att pausas anger du egenskapen `UIApplicationExitsOnSuspend` som `YES`:

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

Minimera nedladdningstorleken genom att läsa in externa, resursexklusiva SWF-filer

AIR 3.7

Du kan minimera storleken på den första programnedladdningen genom att förpacka en delmängd av SWF-filerna som används i programmet och läsa in de återstående (resursexklusiva) externa SWF-filerna i miljön med metoden `Loader.load ()`. Om du vill använda den här funktionen måste du förpacka programmet så att ADT flyttar all ActionScript ByteCode (ABC) från de externt inlästa SWF-filerna till huvudprogrammet för SWF och lämnar en SWF-fil som endast innehåller resurser. Detta görs för att anpassa till regelverket för Apple Store som förbjuder att kod laddas ned sedan programmet har installerats.

ADT gör efterföljande för att stödja externt inlästa SWF-filer (kallas även för "strippade" SWF-filer):

- Läser in textfilen som anges i `<iPhone>`-elementets `<externalSwfs>`-delelement för att komma åt den radavgränsade listan med SWF-filer som ska läsas under körningen:

```
<iPhone>
                                        ...

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

- Överför ABC-koden från varje externt inläst SWF-fil till huvudfilen.
- Utelämnar den externt inlästa SWF-filen från .ipa-filen.
- Kopierar de strippade SWF-filerna till `.remoteStrippedSWFs`-katalogen. Du har dessa SWF-filer på en webbserver och programmet läser in dem vid behov under körningen.

Du anger vilka SWF-filer som ska läsas in under körningen genom att ange deras namn, en per rad i en textfil. Se följande exempel:

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

Den angivna filsökvägen är relativ till programbeskrivningsfilen. Dessutom måste du ange dessa SWF-filer som resurser i `adt`-kommandot.

Obs! *Den här funktionen gäller endast för standardförpackning. För snabbpaketering (med till exempel hjälp av tolk, simulator eller felsökning) skapar ADT inte strippade SWF-filer.*

Mer information om den här funktionen, inklusive exempelkod, finns i External hosting of secondary SWFs for AIR apps on iOS, en blogg som drivs av Adobe-teknikern Abhinav Dhandh.

Geolocation-stöd

För Geolocation-stöd ska du lägga till ett av följande nyckelvärdepar i elementet `InfoAdditions`:

```
<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>
```

Programikoner

Följande tabell visar de ikonstorlekar som används på olika mobilplattformar:

Ikonstorlek	Plattform
29x29	iOS
36x36	Android
40x40	iOS

Ikonstorlek	Plattform
48 x 48	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
120 x 120	iOS
144x144	Android, iOS
152x152	iOS
167 x 167	iOS
180x180	iOS
192 x 192	Android
512 x 512	Android, iOS
1 024 x 1 024	iOS

Ange sökvägen till ikonfilerna i icon-elementet i programbeskrivningsfilen:

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

Om du inte anger någon ikon med en bestämd storlek används den näststörsta storleken, som skalas för att passa.

Ikoner på Android

På Android används de ikoner som angetts i programbeskrivningen som programstartsikoner. Programstartsikonen bör anges som en uppsättning PNG-bilder med 36x36, 48x48, 72x72, 96x96, 144x144 och 192x192 pixlar. De här ikonstorlekarna används för skärmar med låg täthet, mellanhög täthet och hög täthet.

Utvecklarna måste skicka ikonen med 512x512 pixlar när appen överförs till Google Play Store.

Ikoner på iOS

De ikoner som definieras i programbeskrivningen används på följande platser i ett iOS-program:

- En ikon med storleken 29x29 pixlar – Spotlight-sökikon för iPhone- och iPod-enheter med lägre upplösning, och Inställningar-ikon för iPad-enheter med lägre upplösning.
- En ikon med storleken 40 x 40 pixlar – Spotlight-sökningsikon på iPad-enheter med lägre upplösning.
- En ikon med storleken 48x48 pixlar – AIR lägger till en ram i den här bilden och använder den som 50x50-ikon för Spotlight-sökningar på iPad-enheter med lägre upplösning.
- En ikon med storleken 50x50 pixlar – Spotlight-sökning på iPad-enheter med lägre upplösning.
- En ikon med storleken 57x57 pixlar – Programikon för iPhone- och iPod-enheter med lägre upplösning.
- En ikon med storleken 58x58 pixlar – Spotlight-ikon för iPhone- och iPod-enheter med Retina-skärm och Inställningar-ikon för iPad-enheter med Retina-skärm.
- En ikon med storleken 60x60 pixlar – Programikon för iPhone- och iPod-enheter med lägre upplösning.
- En ikon med storleken 72x72 pixlar (valfritt) – Programikon för iPad-enheter med lägre upplösning.
- En ikon med storleken 76x76 pixlar (valfritt) – Programikon för iPad-enheter med lägre upplösning.
- En ikon med storleken 80 x 80 pixlar – Spotlight-sökning på iPhone/iPod/iPad-enheter med hög upplösning.
- En ikon med storleken 100x100 pixlar – Spotlight-sökning på iPad-enheter med Retina-skärm.
- En ikon med storleken 114x114 pixlar – Programikon för iPhone- och iPod-enheter med Retina-skärm.
- En ikon med storleken 120 x 120 pixlar – Programikon på iPhone/iPod-enheter med hög upplösning.
- En ikon med storleken 152 x 152 pixlar – Programikon för iPad-enheter med hög upplösning.
- En ikon med storleken 167 x 167 pixlar – Programikon för iPad Pro-enheter med hög upplösning.
- En ikon med storleken 512x512 pixlar – Programikon för iPhone-, iPod- och iPad-enheter med lägre upplösning. iTunes visar den här ikonen. PNG-filen med 512 pixlar används bara för att testa utvecklingsversioner av ditt program. När du skickar det färdiga programmet till Apple App Store skickar du 512-bilden separat som en JPG-fil. Den ingår inte i IPA-filen.
- En ikon med storleken 1 024x1 024 pixlar – Programikon för iPhone-, iPod- och iPad-enheter med Retina-skärm.

Den glänsande effekten läggs till i iOS. Du behöver inte använda effekten på källbilden. Om du vill ta bort den här glänsande standardeffekten lägger du till elementet `InfoAdditions` i programbeskrivningsfilen:

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

Obs! *I iOS infogas programmetadata som png-metadata i programikoner så att Adobe kan spåra antalet AIR-program som är tillgängliga i Apple iOS app store. Om du inte vill att ditt program ska identifieras som ett AIR-program på grund av dessa ikonmetadata måste du packa upp IPA-filen, ta bort ikonmetadata och paketera om det. Denna procedur beskrivs i artikeln Välja bort AIR-programanalyser för iOS.*

Fler hjälpavsnitt

”icon” på sidan 221

”imageNxN” på sidan 222

För Android-utvecklare: Riktlinjer för ikondesign (på engelska)

iOS Riktlinjer för användargränssnitt: Skapa egna ikoner och bilder

iOS-startbilder

Utöver programikonerna måste du också tillhandahålla minst en startbild med namnet *Default.png*. Om du vill kan du inkludera separata startbilder för olika startorienteringar, upplösningar (inklusive för högupplösta Retina-skärmar och 16:9-förhållandet) och enheter. Du kan också inkludera olika startbilder som ska användas när ditt program anropas via en webbadress.

Startbildsfiler har ingen referens i programbeskrivningen och måste placeras i programmets rotkatalog. (Lägg *inte* filerna i en underkatalog.)

Schema för filnamngivning

Namnge bilderna enligt följande schema:

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

Basnamnsdelen av filnamnet är den enda obligatoriska delen. Den är antingen *Default* (med stort D) eller det namn som du anger med `UILaunchImageFile`-nyckeln i `InfoAdditions`-elementet i programbeskrivningen.

Delen för *skärmstorleksmodifieraren* bestämmer storleken på skärmar när det inte är skärmar med standardstorlek. Denna modifierare används endast för iPhone- och iPod touch-modellerna med skärmar som har storleksförhållandet 16:9, till exempel iPhone 5 och iPod touch (5:e generationen). Det enda tillåtna värdet för den här modifieraren är `-568h`. Eftersom dessa enheter har stöd för högupplösta Retina-skärmar används alltid skärmstorleksmodifieraren även när en bild har skalmodifieraren `@2x`. Den fullständiga standardstartbilden för dessa enheter är `Default-568h@2x.png`.

Delen *urischeme* är den sträng som används för att identifiera URI-schemat. Denna del gäller endast om ditt program har stöd för en eller flera anpassade URL-scheman. Om till exempel ditt program kan anropas via en länk som `example://foo` använder du `-example` i scheme-delen av startbildens filnamn.

Delen *orientation* erbjuder ett sätt att ange olika startbilder som kan användas beroende på enhetens orientering när programmet startas. Denna del gäller endast för bilder i iPad-program. Det kan vara ett av följande värden beroende på den orientering enheten har när programmet startas:

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

Den *skalade* delen är `@2x` (för iPhone 4, iPhone 5 och iPhone 6) eller `@3x` (för iPhone 6 plus) för startbilder som används på skärmar med hög upplösning (retina). (Uteslut delen scale helt för de bilder som används för skärmar med standardupplösning.) Om du vill använda bilder för större enheter, som exempelvis iPhone 5 och iPod touch (5:e generationen), måste du även ange skärmstorleksmodifieraren `-528h` efter basnamnsdelen och före andra delar.

Delen *device* används för att bestämma startbilder för handhållna enheter och telefoner. Denna del används när ditt program är ett generellt program som har stöd för både handhållna enheter och surfplattor med en enda programbinärfil. De möjliga värdena är antingen `~ipad` eller `~iphone` (för både iPhone och iPod Touch).

För iPhone kan du bara inkludera bilder med stående proportioner. För iPhone 6 plus kan även liggande bilder läggas till. Använd bilder med 320x480 pixlar för standardupplösta enheter, bilder med 640x960 pixlar för högupplösta enheter och bilder med 640x1136 pixlar för 16:9-enheter, som exempelvis iPhone 5 och iPod touch (5:e generationen).

På en iPad inkluderar du bilder enligt följande:

- AIR 3.3 och tidigare – Icke-fullstora bilder: Inkludera bilder med både liggande (1 024x748 för normal upplösning, 2 048x1 496 för hög upplösning) och stående (768x1 004 för normal upplösning, 1 536x2 008 för hög upplösning) format.
- AIR 3.4 och senare – Fullstora bilder: Inkludera bilder med både liggande (1 024x768 för normal upplösning, 2 048x1 536 för hög upplösning) och stående (768x1 024 för normal upplösning, 1 536x2 048 för hög upplösning) format. Observera att när du paketerar en fullstor bild för ett program som inte täcker hela skärmen, kommer de 20 översta pixlarna (översta 40 för hög upplösning) att täckas av statusfältet. Undvik att visa viktig information i detta område.

Exempel

I följande tabell visas ett exempel på en uppsättning startbilder som du kan inkludera för ett tänkt program, som har stöd för många typer av enheter och orienteringar och som kan startas via webbadresser med `example://`-schemat:

Filnamn	**Bildstorlek**	**Användning**
Default.png	320 x 480	iPhone, standardupplösning
Default@2x.png	640 x 960	iPhone, hög upplösning
Default-568h@2x.png	640 x 1136	iPhone, hög upplösning, 16:9-förhållande
Default-Portrait.png	768 x 1 004 (AIR 3.3 och tidigare) 768 x 1024 (AIR 3.4 och senare)	iPad, stående orientering
Default-Portrait@2x.png	1 536 x 2 008 (AIR 3.3 och tidigare) 1536 x 2048 (AIR 3.4 och senare)	iPad, hög upplösning, stående orientering
Default-PortraitUpsideDown.png	768 x 1 004 (AIR 3.3 och tidigare)768 x 1024 (AIR 3.4 och senare)	iPad, stående orientering upp och ned
Default-PortraitUpsideDown@2x.png	1 536 x 2 008 (AIR 3.3 och tidigare)1536 x 2048 (AIR 3.4 och senare)	iPad, hög upplösning, stående orientering upp och ned
Default-Landscape.png	1 024 x 768	iPad, liggande orientering åt vänster
Default-LandscapeLeft@2x.png	2048 x 1536	iPad, hög upplösning, liggande orientering vänster
Default-LandscapeRight.png	1 024 x 768	iPad, liggande orientering åt höger
Default-LandscapeRight@2x.png	2048 x 1536	iPad, hög upplösning, liggande orientering höger
Default-example.png	320 x 480	example:// URL på standard-iPhone
Default-example@2x.png	640 x 960	example:// URL på iPhone med hög upplösning
Default-example~ipad.png	768 x 1 004	example:// URL på iPad med stående orienteringar
Default-example-Landscape.png	1 024 x 768	example:// URL på iPad med liggande orienteringar

Detta exempel visar endast ett tillvägagångssätt. Du kan till exempel använda bilden `Default.png` för iPad och ange speciella startbilder för iPhone och iPod med `Default~iphone.png` och `Default@2x~iphone.png`.

Se även

iOS Application Programming Guide: Application Launch Images

Startbilder som ska förpackas för iOS-enheter

Enheter	Upplösning (pixlar)	Startbildens namn	Orientering
iPhone			
iPhone 4 (utan retina)	640 x 960	Default~iphone.png	Stående
iPhone 4, 4s	640 x 960	Default@2x~iphone.png	Stående
iPhone 5, 5c, 5s	640 x 1136	Default-568h@2x~iphone.png	Stående
iPhone6, iPhone7	750 x 1334	Default-375w-667h@2x~iphone.png	Stående
iPhone6+, iPhone 7+	1242 x 2208	Default-414w-736h@3x~iphone.png	Stående
iPhone6+, iPhone7+	2208 x 1242	Default-Landscape-414w-736h@3x~iphone.png	Landscape
iPad			
iPad 1, 2	768 x 1 024	Default-Portrait~ipad.png	Stående
iPad 1, 2	768 x 1 024	Default-PortraitUpsideDown~ipad.png	Stående upp och ned
iPad 1, 2	1 024 x 768	Default-Landscape~ipad.png	Liggande vänster
iPad 1, 2	1 024 x 768	Default-LandscapeRight~ipad.png	Liggande höger
iPad 3, Air	1 536 x 2 048	Default-Portrait@2x~ipad.png	Stående
iPad 3, Air	1 536 x 2 048	Default-PortraitUpsideDown@2x~ipad.png	Stående upp och ned
iPad 3, Air	2048 x 1536	Default-LandscapeLeft@2x~ipad.png	Liggande vänster
iPad 3, Air	2048 x 1536	Default-LandscapeRight@2x~ipad.png	Liggande höger
iPad Pro	2048 x 2732	Default-Portrait@2x.png	Portrait
iPad Pro	2732 x 2048	Default-Landscape@2x.png	Landscape

Riktlinjer för grafik

Du kan skapa vilken grafik du vill för en startbild, förutsatt att grafiken har rätt dimensioner. Men det är oftast bäst att bilden matchar det inledande tillståndet för programmet. Du kan skapa en sådan startbild genom att ta en skärmbild av ditt programs startskärm:

1. Öppna programmet på iOS-enheten. När den första skärmen med användargränssnitt visas trycker du och håller ned hemknappen (nedanför skärmen). Samtidigt som du håller ned hemknappen trycker du på viloläge-/strömknappen (på ovansidan av enheten). Då tas en skärmavbild som skickas till kamerarullen.
2. Överför bilden till utvecklingsdatorn med iPhoto eller med något annat bildöverföringsprogram.

Lägg inte in någon text i startbilden om programmet översätts till flera språk. Startbilden är statisk, och texten kan inte anpassas för andra språk.

Se även
iOS Human Interface Guidelines: Launch images

Ignorerade inställningar

Program på mobilenheter ignorerar programinställningar som gäller funktioner för systemfönster eller operativsystem på stationära datorer. De inställningar som ignoreras är:

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

Paketera AIR-mobilprogram

Använd ADT-kommandot -package för att skapa programpaketet för ett AIR-program som är avsett för en mobilenhet. Parametern -target anger den mobilplattform för vilken paketet skapas.

Android-paket
AIR-program på Android använder Android-paketformatet (APK) i stället för AIR-paketformatet.

Paket som skapas med ADT och måltypen *APK* har ett format som kan skickas till Android Market. Android Market har vissa krav som inskickade program måste uppfylla för att accepteras. Du bör läsa igenom de senaste kraven innan du skapar det slutliga paketet. Läs mer i För Android-utvecklare: Publicera via Market (på engelska).

Till skillnad från iOS-program kan du använda ett vanligt AIR-kodsigneringscertifikat för att signera Android-program, men om du ska skicka det till Android Market måste certifikatet följa Market-reglerna, som kräver att certifikatet är giltigt till minst år 2033. Du kan skapa ett sådant certifikat med ADT-kommandot -certificate.

Om du vill skicka ett program till en alternativ plats, som inte tillåter att programmet hämtar en AIR-fil från Google Market, kan du ange en alternativ hämtningsadress med ADT-parametern `-airDownloadURL`. När en användare som inte har den version av AIR-miljön som krävs startar ditt program dirigeras de om till den angivna webbadressen. Avsnittet ”ADT-kommandot package” på sidan 163 innehåller mer information.

ADT förpackar som standard Android-program med en delad körmiljö. För att köra programmet ska användaren installera den separata körmiljön AIR på sin enhet.

Obs! *Om du vill tvinga ADT att skapa en APK som använder den låsta körmiljön ska du använda* `target apk-captive-runtime`.

iOS-paket

AIR-program på iOS använder iOS-paketformatet (IPK) i stället för AIR-paketformatet.

Paket som skapas med ADT och måltypen *ipa-app-store*, samt korrekt kodsigneringscertifikat och provisioneringsprofil, har ett format som kan skickas till Apple App Store. Använd måltypen *ipa-ad-hoc* för att paketera ett program för ad hoc-distribution.

Du måste använda korrekt utvecklarcertifikat från Apple för att signera programmet. De certifikat som används för att skapa testversioner och de som används för det slutliga paketet som ska skickas in skiljer sig åt.

Mer information om hur ett iOS-program ska paketeras med Ant finns i Piotr Walczyszyn: Packaging AIR application for iOS devices with ADT command and ANT script

Paketera med ADT

I AIR SDK 2.6 och senare finns stöd för paketering för både iOS och Android. Innan du paketerar programmet måste du kompilera all ActionScript-kod, MXML-kod och tilläggskod. Du måste också ha ett kodsigneringscertifikat.

Du hittar detaljerad referensinformation om ADT-kommandon och -alternativ i ”AIR Developer Tool (ADT)” på sidan 162.

Android APK-paket

Skapa ett APK-paket

Om du vill skapa ett APK-paket använder du ADT-kommandot package och anger måltypen som *apk* för officiella versioner, *apk-debug* för felsökningsversioner eller *apk-emulator* för officiella versioner som ska köras på en emulator.

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

Skriv hela kommandot på en rad. Radbrytningarna i ovanstående exempel är bara för att det ska vara lättare att läsa. Exemplet förutsätter även att sökvägen till ADT-verktyget finns i kommandoradens sökvägsdefinition. (Läs ”Systemvariabeln path” på sidan 299 om du behöver mer hjälp.)

Du måste köra kommandot från den katalog som innehåller programfilerna. Programfilerna i exemplet är myApp-app.xml (programbeskrivningsfilen), myApp.swf och en ikonkatalog.

När du kör kommandot så som visas tillfrågas du om lösenordet för nyckelbehållaren i ADT. (De lösenordstecken du skriver visas inte. Tryck på Retur när du är klar.)

***Obs!** Som standard har alla AIR Android-program `air.`-prefixet i paketets namn. Ställ in systemvariabeln `AIR_NOANDROIDFLAIR` på true på datorn om du vill ändra standardbeteendet.*

Skapa ett APK-paket för ett program som använder ANE-tillägg

Om du vill skapa ett APK-paket för ett program där ANE-tillägg används, lägger du till `-extdir`-alternativet förutom de normala paketeringsalternativen. Om flera ANE delar resurser/bibliotek väljer ADT bara en enda resurs/bibliotek och ignorerar andra dubblettposter innan en varning aktiveras. Med detta alternativ anges katalogen som innehåller ANE-filerna som används i programmet. Till exempel:

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

Skapa ett APK-paket som innehåller en egen version av AIR-miljön

Om du vill skapa ett APK-paket som innehåller både program och en låst version av AIR-miljön, ska du använda målet `apk-captive-runtime`. Med detta alternativ anges katalogen som innehåller ANE-filerna som används i programmet. Till exempel:

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

Tänkbara nackdelar på denna teknik är:

- Allvarliga säkerhetskorrigeringar är inte automatiskt tillgängliga för användare när Adobe publicerar en säkerhetsuppdatering
- Större utrymme för program på disken krävs

***Obs!** När du paketerar miljön läggs behörigheterna `INTERNET` och `BROADCAST_STICKY` automatiskt till i ditt program. Dessa behörigheter är obligatoriska för AIR-miljön.*

Skapa ett APK-felsökningspaket

Om du vill skapa en version av programmet som du kan använda med en felsökare använder du apk-debug som mål och anger anslutningsalternativ:

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

Flaggan -connect talar om för AIR-miljön på enheten var i nätverket fjärrfelsökaren finns. Om du vill felsöka via USB måste du ange flaggan `-listen` i stället och ange den TCP-port som ska användas för felsökningsanslutningen:

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

De flesta felsökningsfunktioner kräver dessutom att du kompilerar programmets SWF- och SWC-filer med felsökning aktiverat. Avsnittet ”Anslutningsalternativ för felsökning” på sidan 179 innehåller en fullständig beskrivning av flaggorna `-connect` och `-listen`.

Obs! *I ADT förpackas som standard en låst kopia av AIR-miljön tillsammans med din Android-app medan appen förpackas med målet apk-debug. Om du vill framtvinga ADT att skapa en APK, som använder en intern miljö, anger du att miljövariabeln* `AIR_ANDROID_SHARED_RUNTIME` *ska vara* `true`*.*

På Android måste programmet även ha behörighet att komma åt Internet för att ansluta till den dator som kör felsökaren i nätverket. Läs mer i ”Android-behörigheter” på sidan 74.

Skapa ett APK-paket som ska användas på en Android-emulator

Du kan använda ett APK-felsökningspaket på en Android-emulator, men inte ett paket med en officiell version. Om du vill skapa ett officiellt APK-paket som ska användas på en emulator använder du ADT-kommandot package och anger måltypen som *apk-emulator*:

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

Exemplet förutsätter att sökvägen till ADT-verktyget finns i kommandoradens sökvägsdefinition. (Läs ”Systemvariabeln path” på sidan 299 om du behöver mer hjälp.)

Skapa ett APK-paket från en AIR- eller AIRI-fil

Du kan skapa ett APK-paket direkt från en befintlig AIR- eller AIRI-fil:

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

AIR-filen måste använda AIR 2.5-namnutrymmet (eller senare) i programbeskrivningsfilen.

Skapa ett APK-paket för Android x86-plattformen

Från och med AIR 14, kan argumentet `-arch` användas för att förpacka en APK för Android x86-plattformen. Till exempel:

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

iOS-paket

På iOS konverteras SWF-filens bytekod och andra källfiler till ett systemspecifikt iOS-program.

1. Skapa SWF-filen med Flash Builder, Flash Professional eller en kommandoradskompilerare.
2. Öppna kommandoskalet eller terminalen och gå till projektmappen för iPhone-programmet.

3 Använd sedan ADT-verktyget för att skapa IPA-filen med följande syntax:

```
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
```

Ändra referensen `adt` så att hela sökvägen till ADT-programmet finns med. ADT installeras i underkatalogen bin för AIR SDK.

Välj det `-target`-alternativ som bäst stämmer överens med den typ av iPhone-program som du vill skapa:

- `-target ipa-test` – Välj det här alternativet om du snabbt vill kompilera en version av programmet för att testa det på din utvecklar-iPhone. Du kan även använda `ipa-test-interpreter` för snabbare kompilering eller `ipa-test-interpreter-simulator` för att köra iOS-simulatorn.
- `-target ipa-debug` – Välj det här alternativet om du vill kompilera en felsökningsversion av programmet för att testa det på din utvecklar-iPhone. Med det här alternativet kan du använda en felsökningssession för att ta emot `trace()`-utdata från iPhone-programmet.

  Du kan inkludera ett av följande `-connect`-alternativ (`CONNECT_OPTIONS`) för att ange IP-adressen till den utvecklingsdator som felsökaren körs på:

  - `-connect` – Programmet försöker via wifi ansluta till en felsökningssession på den utvecklingsdator som används för att kompilera programmet.
  - `-connect IP_ADDRESS` – Programmet försöker via wifi ansluta till en felsökningssession på datorn med den angivna IP-adressen. Till exempel:

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

  - `-connect HOST_NAME` – Programmet försöker via wifi ansluta till en felsökningssession på datorn med det angivna värdnamnet. Till exempel:

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

  Alternativet `-connect` är valfritt. Om det inte anges försöker det skapade felsökningsprogrammet inte att ansluta till någon felsökningssession. Du kan ange `-listen` i stället för `-connect` för att aktivera USB-felsökning. Se ”Fjärrfelsökning med FDB via USB” på sidan 104.

  Om ett anslutningsförsök till en felsökningssession misslyckas visas en dialogruta i programmet, där användaren uppmanas ange IP-adressen till felsökningsvärden. Ett anslutningsförsök kan misslyckas om enheten inte är ansluten till wifi. Det kan också inträffa om enheten är ansluten, men inte finnas innanför felsökningsvärdens brandvägg.

  Du kan även använda `ipa-debug-interpreter` för snabbare kompilering eller `ipa-debug-interpreter-simulator` för att köra iOS-simulatorn.

  Du hittar mer information i ”Felsöka AIR-mobilprogram” på sidan 98.

- `-target ipa-ad-hoc` – Välj det här alternativet om du vill skapa ett program för ad hoc-distribution. Se Apples utvecklingscenter för iPhone

- `-target ipa-app-store` – Välj det här alternativet om du vill skapa en slutgiltig version av IPA-filen för distribution till Apple App Store.

Ersätt *`PROFILE_PATH`* med sökvägen till filen för provisioneringsprofilen för ditt program. Mer information om provisioneringsprofiler finns i ”iOS-konfiguration” på sidan 64.

Använd alternativet `-platformsdk` för att peka på SKD:n för iOS-simulatorn när du skapar ett program för iOS-simulatorn.

Ersätt *`SIGNING_OPTIONS`* med en referens till ditt iPhone-utvecklarcertifikat och lösenord. Använd följande syntax:

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

Ersätt *P12_FILE_PATH* med sökvägen till P12-certifikatfilen. Ersätt *PASSWORD* med certifikatlösenordet. (Se exemplet nedan.) Du hittar mer information om P12-certifikatfilen i ”Konvertera ett utvecklarcertifikat till en P12-fil” på sidan 194.

Obs! *Du kan använda ett självsignerat certifikat när du paketerar för iOS-simulatorn.*

Ersätt *`APP_DESCRIPTOR`* med en referens till programbeskrivningsfilen.

Ersätt *`SOURCE_FILES`* med en referens till SWF-huvudfilen för ditt projekt följt av alla eventuella andra resurser som du vill ha med. Inkludera sökvägarna till alla ikonfiler som du definierade i dialogrutan för programinställningarna i Flash Professional eller i en egen programbeskrivningsfil. Lägg dessutom till den inledande skärmbildsgrafiken med filen Default.png.

Använd alternativet `-extdir` *`extension-directory`* för att ange katalogen som innehåller ANE-filerna (systemspecifika tillägg) som används i programmet. Inkludera inte detta alternativ om inga ANE-tillägg används i programmet.

Viktigt! *Skapa inte en underkatalog i programkatalogen med namnet `Resources`. En mapp med detta namn skapas automatiskt i miljön för att uppfylla villkoren för IPA-paketstrukturen. Om du skapar en egen Resources-mapp kommer det att leda till en allvarlig konflikt.*

Skapa ett iOS-paket för felsökning

Om du vill skapa ett iOS-paket som ska installeras på testenheter använder du ADT-kommandot package och anger måltypen som *ios-debug*. Innan du kör det här kommandot måste du ha fått ett kodsigneringscertifikat för utvecklare och en provisioneringsprofil av Apple.

```
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
```

Obs! *Du kan även använda `ipa-debug-interpreter` för snabbare kompilering eller `ipa-debug-interpreter-simulator` för att köra iOS-simulatorn*

Skriv hela kommandot på en rad. Radbrytningarna i ovanstående exempel är bara för att det ska vara lättare att läsa. Exemplet förutsätter även att sökvägen till ADT-verktyget finns i kommandoradens sökvägsdefinition. (Läs ”Systemvariabeln path” på sidan 299 om du behöver mer hjälp.)

Du måste köra kommandot från den katalog som innehåller programfilerna. Programfilerna i exemplet är myApp-app.xml (programbeskrivningsfilen), myApp.swf, en ikonkatalog och filen Default.png.

Du måste signera programmet med rätt distributionscertifikat från Apple. Du kan inte använda några andra kodsigneringscertifikat.

Använd alternativet `-connect` för wifi-felsökning. Programmet försöker initiera en felsökningssession med Flash Debugger (FDB), som körs på den angivna IP-adressen eller det angivna värdnamnet. Använd alternativet `-listen` för USB-felsökning. Du börjar med att starta programmet och sedan FDB, vilket initierar en felsökningssession för det program som körs. Läs ”Ansluta till Flash Debugger” på sidan 102 om du vill veta mer.

Skapa ett iOS-paket för Apple App Store-distribution

Om du vill skapa ett iOS-paket som ska skickas till Apple App Store använder du ADT-kommandot package och anger måltypen som *ios-app-store*. Innan du kör det här kommandot måste du ha fått ett kodsigneringscertifikat för distribution och en provisioneringsprofil av Apple.

```
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
```

Skriv hela kommandot på en rad. Radbrytningarna i ovanstående exempel är bara för att det ska vara lättare att läsa. Exemplet förutsätter även att sökvägen till ADT-verktyget finns i kommandoradens sökvägsdefinition. (Läs ”Systemvariabeln path” på sidan 299 om du behöver mer hjälp.)

Du måste köra kommandot från den katalog som innehåller programfilerna. Programfilerna i exemplet är myApp-app.xml (programbeskrivningsfilen), myApp.swf, en ikonkatalog och filen Default.png.

Du måste signera programmet med rätt distributionscertifikat från Apple. Du kan inte använda några andra kodsigneringscertifikat.

Viktigt! *Apple kräver att du använder Apple-programmet Application Loader för att överföra program till deras App Store. Apple publicerar bara Application Loader för Mac OS X. Även om du kan utveckla AIR-program för iPhone med en Windows-dator måste du alltså ha tillgång till en Mac med OS X (version 10.5.3 eller senare) för att kunna skicka programmet till Apple App Store. Du kan hämta Application Loader från Apple iOS Developer Center.*

Skapa ett iOS-paket för ad hoc-distribution

Om du vill skapa ett iOS-paket för ad hoc-distribution använder du ADT-kommandot package och anger måltypen som *ios-ad-hoc*. Innan du kör det här kommandot måste du ha fått korrekt kodsigneringscertifikat för ad hoc-distribution och en provisioneringsprofil av Apple.

```
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
```

Skriv hela kommandot på en rad. Radbrytningarna i ovanstående exempel är bara för att det ska vara lättare att läsa. Exemplet förutsätter även att sökvägen till ADT-verktyget finns i kommandoradens sökvägsdefinition. (Läs ”Systemvariabeln path” på sidan 299 om du behöver mer hjälp.)

Du måste köra kommandot från den katalog som innehåller programfilerna. Programfilerna i exemplet är myApp-app.xml (programbeskrivningsfilen), myApp.swf, en ikonkatalog och filen Default.png.

Du måste signera programmet med rätt distributionscertifikat från Apple. Du kan inte använda några andra kodsigneringscertifikat.

Skapa ett iOS-paket för ett program som använder ANE-tillägg
Om du vill skapa ett iOS-paket för ett program som använder ANE-tillägg ska du använda ADT-paketkommandot med alternativet `-extdir`. Använd det ADT-kommando som är lämpligt för målet (`ipa-app-store`, `ipa-debug`, `ipa-ad-hoc`, `ipa-test`). Till exempel:

```
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
```

Skriv hela kommandot på en rad. Radbrytningarna i ovanstående exempel är bara för att det ska vara lättare att läsa.

Med avseende på ANE-tillägg antas i exemplet att katalogen med namnet `extensionsDir` är den katalog där kommandot ska köras. Katalogen `extensionsDir` innehåller de ANE-filer som används i programmet.

Felsöka AIR-mobilprogram

Du kan felsöka ett AIR-mobilprogram på flera sätt. Det enklaste sättet att hitta problem i programlogiken är att felsöka med ADL eller iOS-simulatorn på utvecklingsdatorn. Du kan också installera ditt program på en enhet och fjärrfelsöka med Flash Debugger på en stationär dator.

Enhetssimulering med ADL

Det snabbaste och enklaste sättet att testa och felsöka de flesta funktioner i mobilprogram är att köra programmet på utvecklingsdatorn med ADL-verktyget (Adobe Debug Launcher). ADL använder elementet `supportedProfiles` i programbeskrivningen för att fastställa vilken profil som ska användas. Om det finns fler än en profil används den första i listan. Du kan också använda parametern `-profile` i ADL för att välja en av de andra profilerna i listan `supportedProfiles`. (Om du inte inkluderar ett `supportedProfiles`-element i programbeskrivningen kan vilken profil som helst anges för argumentet `-profile`.) Använd till exempel följande kommando för att starta ett program för att simulera mobilprofilen:

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

När du simulerar mobilprofilen på skrivbordet på det här sättet körs programmet i en miljö som bättre motsvarar en mobilenhet. ActionScript-API:er som inte ingår i mobilprofilen är inte tillgängliga. ADL ser dock ingen skillnad på olika mobilenheters funktioner. Du kan till exempel skicka simulerade valknappstryckningar till ditt program, även om den faktiska målenheten inte har några valknappar.

ADL har stöd för att simulera när enhetens orientering ändras och för valknappsinmatningar via menykommandon. När du kör ADL i mobilprofilen visas en meny (antingen i programfönstret eller på datorns menyrad) där du kan ange enhetsrotation eller indata för valknappar.

Indata för valknappar
ADL simulerar valknapparna för bakåt-, meny- och sökknapparna på en mobilenhet. Du kan skicka de här knapparna till den simulerade enheten med den meny som visas när ADL startas med mobilprofilen.

Enhetsrotation

Med ADL kan du simulera enhetsrotation via den meny som visas när ADL startas med mobilprofilen. Du kan rotera den simulerade enheten till höger eller vänster.

Rotationssimuleringen påverkar bara program som har automatisk orientering aktiverat. Du kan aktivera den här funktionen genom att ange elementet `autoOrients` som `true` i programbeskrivningen.

Skärmstorlek

Du kan testa programmet med olika skärmstorlekar genom att ange ADL-parametern `-screensize`. Du kan skicka koden för en av de fördefinierade skärmtyperna eller en sträng med fyra värden, som representerar pixeldimensionerna för den normala och den maximerade skärmstorleken.

Ange alltid pixeldimensioner för stående layout, d.v.s. ange bredden som ett värde som är mindre än höjden. Till exempel öppnar följande kommando ADL med en simulering av den skärm som används på Motorola Droid:

```
adl -screensize 480x816:480x854 myApp-app.xml
```

Du hittar en lista över de fördefinierade skärmtyperna i avsnittet ”ADL-användning” på sidan 156.

Begränsningar

Vissa API:er som inte stöds i skrivbordsprofilen kan inte simuleras med ADL. Bland de API:er som inte kan simuleras finns följande:

- Accelerometer
- cacheAsBitmapMatrix
- CameraRoll
- CameraUI
- Geolocation
- Flera beröringar och gester på operativsystem för stationära datorer som saknar stöd för de här funktionerna
- SystemIdleMode

Om ditt program använder de här klasserna bör du testa funktionerna på en verklig enhet eller emulator.

Det finns också API:er som fungerar när ADL körs på skrivbordet, men som inte fungerar på alla typer av mobilenheter. Dessa inkluderar:

- Speex- och AAC-ljudkodekar
- Stöd för hjälpmedel och skärmläsare
- RTMPE
- Inläsning av SWF-filer med ActionScript-bytekod
- PixelBender-skuggningar

Glöm inte att testa program som använder de här funktionerna på målenheterna, eftersom ADL inte reproducerar körningsmiljön exakt.

Enhetssimulering med iOS-simulatorn

iOS-simulatorn (endast Mac) erbjuder snabb körning och felsökning av iOS-program. När du testar iOS-simulatorn behöver du inget utvecklarcertifikat och ingen provisioneringsprofil. Du måste fortfarande skapa ett p12-certifikat, trots att det kan vara självsignerat.

Som standard startar ADT alltid iPhone-simulatorn. Gör följande för att ändra simulatorenhet:

- Använd kommandot nedan för att visa de tillgängliga simulatorerna.

  ```
  xcrun simctl list devices
  ```

  Utdata liknar de som visas nedan.

  ```
  == 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)
  ....
  ```

- Du kan välja en specifik simulator genom att ställa in miljövariabeln `AIR_IOS_SIMULATOR_DEVICE` enligt följande:

  ```
  export AIR_IOS_SIMULATOR_DEVICE = 'iPad Retina'
  ```

Starta om processen när du har ställt in systemvariabeln och kör programmet på simulatorenheten.

Obs! *När du använder ADT med iOS-simulatorn måste du alltid inkludera alternativet `-platformsdk` för att ange sökvägen till SDK:n för iOS-simulatorn.*

Så här kör du ett program i iOS-simulatorn:

1 Använd adt -package-kommandot med antingen `-target ipa-test-interpreter-simulator` eller `-target ipa-debug-interpreter-simulator`, vilket visas i följande exempel:

```
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
```

Obs! *Signeringsalternativ krävs inte längre för simulatorer, därför kan vilket värde som helst anges för `-keystore`-flaggan eftersom den ignoreras av ADT.*

2 Använd adt -installApp-kommandot för att installera programmet i iOS-simulatorn, vilket visas i följande exempel:

```
adt -installApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -package sample_ipa_name.ipa
```

3 Använd adt -launchApp-kommandot för att köra programmet i iOS-simulatorn, vilket visas i följande exempel:

Obs! *Som standard kör kommandot* `adt -launchApp` *programmet i iPhone-simulatorn. Om du vill köra programmet i iPad-simulatorn ska du exportera miljövariabeln* `AIR_IOS_SIMULATOR_DEVICE` *= "iPad" och sedan använda kommandot* `adt - launchApp`*.*

```
adt -launchApp
                                -platform ios
                                -platformsdk
/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator5.0.sdk
                                -device ios-simulator
                                -appid sample_ipa_name
```

Om du vill testa ett ANE-tillägg i iOS-simulatorn ska du använda `iPhone-x86` som plattformsnamn i filen extension.xml och ange `library.a` (statiskt bibliotek) i elementet `nativeLibrary`, vilket visas i följande extension.xml-exempel:

```
<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>
```

Obs! *När du testar ett ANE-tillägg i iOS-simulatorn ska du inte använda det statiska biblioteket (*`.a`*-fil) som är kompilerat för enheten. Använd i stället det statiska biblioteket som är kompilerat för simulatorn.*

Trace-programsatser

När du kör mobilprogrammet på skrivbordet skrivs utdata från trace-programsatser antingen till felsökaren eller till det terminalfönster som användes för att starta ADL. När du kör programmet på en enhet eller en emulator kan du konfigurera en fjärrfelsökningssession för att visa utdata från trace-programsatser. Om det stöds kan du även visa utdata från trace-programsatser med de programvaruutvecklingsverktyg som tillverkaren av enheten eller operativsystemet tillhandahåller.

SWF-filerna i programmet måste i vilket fall som helst kompileras med felsökning aktiverad för att miljön ska kunna generera några trace-programsatser.

Externa trace-programsatser på Android

På en Android-enhet eller -emulator kan du visa utdata från trace-programsatser i systemets logg med hjälp av verktyget Android Debug Bridge (ADB), som ingår i Android SDK. Du visar utdata från ditt program genom att köra följande kommando från en kommandotolk eller ett terminalfönster på utvecklingsdatorn:

```
tools/adb logcat air.MyApp:I *:S
```

där *MyApp* är ditt programs program-ID i AIR. Argumentet `*:S` undertrycker utdata från alla andra processer. Om du vill visa systeminformation utöver trace-utdata för ditt program kan du inkludera ActivityManager i specifikationen för logcat-filtret:

```
tools/adb logcat air.MyApp:I ActivityManager:I *:S
```

De här kommandoexemplen förutsätter att du kör ADB från Android SDK-mappen eller att du har lagt till mappen `SDK` i systemvariabeln path.

Obs! *I AIR 2.6+ ingår ADB-verktyget i AIR SDK och finns i mappen lib/android/bin.*

Externa trace-programsatser på iOS

Om du vill visa utdata från trace-programsatser för ett program som körs på en iOS-enhet måste du skapa en fjärrfelsökningssession med FDB (Flash Debugger).

Fler hjälpavsnitt

Android Debug Bridge: Enable logcat Logging (Aktivera logcat-loggning)

”Systemvariabeln path” på sidan 299

Ansluta till Flash Debugger

Om du vill felsöka ett program som körs på en mobilenhet kan du köra Flash Debugger på utvecklingsdatorn och ansluta till den via nätverket. Du måst göra följande för att aktivera fjärrfelsökning:

- På Android anger du behörigheten android:permission.INTERNET i programbeskrivningen.
- Kompilera SWF-programfilerna med felsökning aktiverat.
- Paketera programmet med `-target apk-debug, för Android,` eller `-target ipa-debug`, för iOS, och antingen flaggan `-connect` (wifi-felsökning) eller `-listen` (USB-felsökning).

För fjärrfelsökning via wifi måste enheten ha tillgång till TCP-porten 7935, på den dator där Flash Debugger körs via IP-adressen, eller det fullständiga kvalificerade domännamnet. För fjärrfelsökning med USB måste enheten ha åtkomst till TCP-porten 7936 eller till porten som är angiven i flaggan `-listen`.

För iOS kan du även ange `-target ipa-debug-interpreter` eller `-target ipa-debug-interpreter-simulator`.

Fjärrfelsökning med Flash Professional

När ditt program är klart för felsökning och behörigheterna har angetts i programbeskrivningen gör du följande:

1. Öppna dialogrutan AIR for Android-inställningar.
2. Gå till fliken Distribution:
   - Markera ”Felsökning” som distributionstyp.
   - Markera ”Installera programmet på den anslutna Android-enheten” vid Efter publicering.
   - Avmarkera ”Starta programmet på den anslutna Android-enheten” vid Efter publicering.
   - Ange sökvägen till Android SDK, vid behov.
3. Klicka på Publicera.

   Ditt program installeras och startas på enheten.
4. Stäng dialogrutan AIR for Android-inställningar.
5. Välj Felsök > Starta fjärrfelsökning > ActionScript 3 på menyn för Flash Professional.

   Meddelandet ”Väntar på att spelaren ska ansluta...” visas på utdatapanelen i Flash Professional.
6. Starta programmet på enheten.
7. Ange IP-adressen eller värdnamnet för den dator på vilken Flash Debugger körs i dialogrutan för anslutning i Adobe AIR. Klicka sedan på OK.

Fjärrfelsökning med FDB via en nätverksanslutning

Om du vill felsöka ett program på en enhet med kommandoradsverktyget Flash Debugger (FDB) kör du först felsökaren på utvecklingsdatorn och startar sedan programmet på enheten. I följande procedur används AMXMLC-, FDB- och ADT-verktygen för att kompilera, paketera och felsöka ett program på enheten. Exemplet förutsätter att du använder en kombination av Flex och AIR SDK och att bin-katalogen ingår i systemvariabeln path. (Detta är bara för att förenkla kommandoexemplen.)

1 Öppna ett terminalfönster eller en kommandotolk och gå till den katalog som innehåller källkoden för programmet.

2 Kompilera programmet med amxmlc och aktivera felsökning:

```
amxmlc -debug DebugExample.as
```

3 Paketera programmet med antingen `apk-debug` eller `ipa-debug` som mål:

```
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
```

Om du alltid använder samma värdnamn eller IP-adress för felsökning kan du ange det värdet efter flaggan `-connect`. Programmet kommer att försöka ansluta till den IP-adressen eller värden automatiskt. I annat fall måste du ange informationen på enheten varje gång du startar felsökningen.

4 Installera programmet.

På Android kan du använda ADT-kommandot `-installApp`:

```
adt -installApp -platform android -package DebugExample.apk
```

På iOS kan du installera programmet med ADT-kommandot `-installApp` eller via iTunes.

5 Öppna ett andra terminalfönster eller en andra kommandotolk och kör FDB:

```
fdb
```

6 Skriv kommandot `run` i FDB-fönstret:

```
Adobe fdb (Flash Player Debugger) [build 14159]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
                                    Waiting for Player to connect
```

7 Starta programmet på enheten.

8 När programmet startas på enheten eller emulatorn öppnas dialogrutan för anslutning i Adobe AIR. (Om du angav värdnamnet eller IP-adressen med alternativet -connect när du paketerade programmet försöker det ansluta automatiskt med den adressen.) Ange korrekt adress och knacka på OK.

För att kunna ansluta till felsökaren i det här läget måste enheten kunna tolka adressen eller värdnamnet och ansluta till TCP-port 7935. Det krävs en nätverksanslutning.

9 När fjärrmiljön ansluter till felsökaren kan du ange brytpunkter med FDB-kommandot `break` och sedan starta körningen med kommandot `continue`:

```
(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
```

Fjärrfelsökning med FDB via USB

AIR 2.6 (Android) AIR 3.3 (iOS)

Om du vill felsöka ett program via en USB-anslutning kan du paketera programmet med alternativet `-listen` i stället för alternativet `-connect`. När du anger alternativet `-listen` avlyssnas i miljön om det finns en anslutning från Flash Debugger (FDB) på TCP-porten 7936 när du startar programmet. Du kör sedan FDB med alternativet `-p` och FDB kommer att initiera anslutningen.

USB-felsökning för Android

För att Flash Debugger, som körs på den stationära datorn, ska kunna ansluta till AIR-miljön som körs på enheten eller emulatorn, måste du använda verktygen Android Debug Bridge (ADB, från Android SDK) eller iOS Debug Bridge (IDB, från AIR SDK) för att vidarebefordra enhetsporten till datorporten.

1. Öppna ett terminalfönster eller en kommandotolk och gå till den katalog som innehåller källkoden för programmet.
2. Kompilera programmet med amxmlc och aktivera felsökning:

   ```
   amxmlc -debug DebugExample.as
   ```

3. Paketera programmet med rätt felsökningsmål (till exempel `apk-debug`) och ange alternativet `-listen`:

   ```
   adt -package -target apk-debug -listen -storetype pkcs12 -keystore ../../AndroidCert.p12
   DebugExample.apk DebugExample-app.xml DebugExample.swf
   ```

4. Anslut enheten till felsökningsdatorn med en USB-kabel. (Du kan också använda den här proceduren för att felsöka ett program som körs på en emulator, och i så fall är en USB-anslutning inte nödvändig – eller ens möjlig).
5. Installera programmet.

   Du kan använda ADT-kommandot `-installApp`:

   ```
   adt -installApp -platform android -package DebugExample.apk
   ```

6. Vidarebefordra TCP-port 7936 från enheten eller emulatorn till den stationära datorn med Android-verktyget ADB:

   ```
   adb forward tcp:7936 tcp:7936
   ```

7. Starta programmet på enheten.
8. Kör FDB med alternativet -p i ett terminalfönster eller en kommandotolk:

   ```
   fdb -p 7936
   ```

9. Skriv kommandot `run` i FDB-fönstret:

   ```
   Adobe fdb (Flash Player Debugger) [build 14159]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
   ```

10 FDB-verktyget försöker ansluta till programmet.

11 När fjärranslutningen har skapats kan du ange brytpunkter med FDB-kommandot `break` och sedan starta körningen med kommandot `continue`:

```
(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
```

Obs! *Portnummer 7936 används som standard för USB-felsökning av både AIR-miljön och av FDB. Du kan ange olika portar för användning med ADT-portparametern -listen och FDB-portparametern -p. I så fall måste du använda verktyget Android Debug Bridge för att vidarebefordra det portnummer som angetts i ADT till den port som angetts i FDB:* `adb forward tcp:adt_listen_port# tcp:fdb_port#`

USB-felsökning för iOS

För att Flash Debugger som körs på den stationära datorn ska kunna ansluta till AIR-miljön, som körs på enheten eller emulatorn, måste du använda verktyget iOS Debug Bridge (IDB, från AIR SDK) för att vidarebefordra enhetsporten till datorporten.

1 Öppna ett terminalfönster eller en kommandotolk och gå till den katalog som innehåller källkoden för programmet.

2 Kompilera programmet med amxmlc och aktivera felsökning:

```
amxmlc -debug DebugExample.as
```

3 Paketera programmet med rätt felsökningsmål (till exempel `ipa-debug` eller `ipa-debug-interpreter`) och ange alternativet `-listen`:

```
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 Anslut enheten till felsökningsdatorn med en USB-kabel. (Du kan också använda den här proceduren för att felsöka ett program som körs på en emulator, och i så fall är en USB-anslutning inte nödvändig – eller ens möjlig).

5 Installera och kör programmet på iOS-enheten. I AIR 3.4 och senare kan du använda `adt -installApp` för att installera programmet via USB.

6 Bestäm enhetens referens (handle) genom att använda IDB-kommandot `-devices` (IDB finns i *`air_sdk_root`*`/lib/aot/bin/iOSBin/idb`):

```
./idb -devices
                                    List of attached devices
                                    Handle    UUID
                                        1     91770d8381d12644df91fbcee1c5bbdacb735500
```

Obs! *(AIR 3.4 och senare) Du kan använda* `adt -devices` *i stället för* `idb -devices` *för att bestämma enhetens referens (handle).*

7 Vidarebefordra en port på datorn till porten som är angiven i parametern `adt -listen` (i detta exempel 16000; standard är 7936) med IDB-verktyget och det enhets-ID som hittades i föregående steg:

```
idb -forward 7936 16000 1
```

I detta exempel är 7936 porten på datorn, 16000 är porten som den anslutna enheten avlyssnar och 1 är enhets-ID för den anslutna enheten.

8 Kör FDB med alternativet -p i ett terminalfönster eller en kommandotolk:

```
fdb -p 7936
```

9 Skriv kommandot `run` i FDB-fönstret:

```
Adobe fdb (Flash Player Debugger) [build 23201]
                                    Copyright (c) 2004-2007 Adobe, Inc. All rights reserved.
                                    (fdb) run
```

10 FDB-verktyget försöker ansluta till programmet.

11 När fjärranslutningen har skapats kan du ange brytpunkter med FDB-kommandot `break` och sedan starta körningen med kommandot `continue`:

Obs! *Portnummer 7936 används som standard för USB-felsökning av både AIR-miljön och av FDB. Du kan ange olika portar för användning med IDB-portparametern -listen och FDB-portparametern -p.*

Installera AIR och AIR-program på mobilenheter

Ditt programs slutanvändare kan installera AIR-miljön och AIR-program med hjälp av enhetens vanliga funktioner för detta.

På Android kan användarna till exempel installera program från Android Market. Om de har tillåtit installation av program från okända källor i programinställningarna kan de även installera ett program genom att klicka på en länk på en webbsida eller genom att kopiera ett programpaket till enheten och öppna det. Om en användare försöker installera ett Android-program, men inte har installerat AIR-miljön än, dirigeras de automatiskt till Market, där de kan installera miljön.

På iOS finns det två sätt att distribuera program till slutanvändare. Den främsta distributionskanalen är Apple App Store. Du kan också använda ad hoc-distribution för att låta ett begränsat antal användare installera ditt program utan att gå via Apple App Store.

Installera AIR-miljön och AIR-program för utveckling

Eftersom AIR-program på mobilenheter installeras som systemspecifika paket kan du använda de vanliga plattformsfunktionerna för att installera program för testning. Om det stöds kan du använda ADT-kommandon för att installera AIR-miljön och AIR-program. Detta stöds för tillfället på Android.

På iOS kan du installera program för testning med iTunes. Testprogram måste signeras med ett kodsigneringscertifikat från Apple, som utfärdats särskilt för programutveckling, och paketeras med en provisioneringsprofil för utveckling. Ett AIR-program är ett självständigt paket på iOS. Ingen separat körningsmiljö används.

Installera AIR-program med ADT

När du utvecklar AIR-program kan du använda ADT för att installera och avinstallera både miljön och dina program. (Den integrerade utvecklingsmiljön kan även innehålla dessa kommandon, och då behöver du inte köra ADT själv.)

Du kan installera AIR-miljön på en enhet eller emulator med ADT-verktyget i AIR. Den SDK som gäller för enheten måste vara installerad. Använd kommandot `-installRuntime`:

```
adt -installRuntime -platform android -device deviceID -package path-to-runtime
```

Om parametern `-package` inte anges väljs lämpligt miljöpaket för enheten eller emulatorn från de som är tillgängliga i den installerade versionen av AIR SDK.

Om du vill installera ett AIR-program på Android eller iOS (AIR 3.4 eller senare) använder du kommandot `-installApp`:

```
adt -installApp -platform android -device deviceID -package path-to-app
```

Det värde som anges för argumentet `-platform` bör matcha den enhet på vilken du installerar.

Obs! *Befintliga versioner av AIR-miljön eller AIR-programmet måste tas bort innan du installerar om.*

Installera AIR-program på iOS-enheter via iTunes

Så här installerar du ett AIR-program på en iOS-enhet för testning:

1. Öppna iTunes-programmet.
2. Om du inte redan har gjort det ska du lägga till provisioneringsprofilen för programmet i iTunes. Välj Arkiv > Lägg till i biblioteket i iTunes. Välj sedan filen med provisioneringsprofilen (med filtypen mobileprovision).
3. I vissa versioner av iTunes ersätts inte programmet om samma version av programmet redan finns installerat. I så fall tar du bort programmet från din enhet och från listan över program i iTunes.
4. Dubbelklicka på IPA-filen för programmet. Den bör visas i listan med program på iTunes.
5. Anslut enheten till USB-porten på datorn.
6. Kontrollera programfliken för enheten i iTunes och att programmet är markerat i listan över program som ska installeras.
7. Markera enheten i den vänstra listan i iTunes-programmet. Klicka sedan på Synkronisera. När synkroniseringen är slutförd visas programmet Hello World i din iPhone.

Om den nya versionen inte har installerats tar du bort programmet från enheten och från listan över program i iTunes och upprepar sedan proceduren. Detta kan bero på att den installerade versionen använder samma program-ID och version.

Fler hjälpavsnitt

”ADT-kommandot installRuntime” på sidan 174

”ADT-kommandot installApp” på sidan 171

Köra AIR-program på en enhet

Du kan starta installerade AIR-program med enhetens användargränssnitt. Om det stöds kan du även fjärrstarta program med ADT-verktyget i AIR:

```
adt -launchApp -platform android -device deviceID -appid applicationID
```

Värdet för argumentet `-appid` måste vara AIR-program-ID:t för det AIR-program som ska startas. Använd det värde som anges i AIR-programbeskrivningen (utan det *air.*-prefix som läggs till under paketering).

Om bara en enhet eller emulator är ansluten och igång kan du utelämna flaggan `-device`. Det värde som anges för argumentet `-platform` bör matcha den enhet på vilken du installerar. För tillfället är *android* det enda värde som stöds.

Ta bort AIR-miljön och AIR-program

Du kan använda enhetens vanliga funktioner för borttagning av program. Om det stöds kan du även använda ADT-verktyget i AIR för att ta bort AIR-miljön och AIR-program. Ta bort miljön med kommandot `-uninstallRuntime`:

```
adt -uninstallRuntime -platform android -device deviceID
```

Avinstallera ett program med kommandot `-uninstallApp`:

```
adt -uninstallApp -platform android -device deviceID -appid applicationID
```

Om bara en enhet eller emulator är ansluten och igång kan du utelämna flaggan `-device`. Det värde som anges för argumentet `-platform` bör matcha den enhet på vilken du installerar. För tillfället är *android* det enda värde som stöds.

Konfigurera en emulator

Om du vill köra ditt AIR-program på en enhetsemulator måste du vanligen använda enhetens SDK för att skapa och köra en emulatorinstans på utvecklingsdatorn. Sedan kan du installera emulatorversionen av AIR-miljön och ditt AIR-program på emulatorn. Tänk på att program på en emulator oftast körs betydligt långsammare än på en riktig enhet.

Skapa en Android-emulator

1. Starta Android SDK och AVD Manager-programmet:
   - På Windows kör du SDK-filen Setup.exe i rotkatalogen för Android SDK.
   - På Mac OS kör du android-programmet i underkatalogen tools i Android SDK-katalogen.
2. Välj alternativet Settings och markera alternativet "Force https://".
3. Markera alternativet Available Packages. Det bör nu visas en lista med tillgängliga SDK:er för Android.
4. Välj en kompatibel Android SDK (Android 2.3 eller senare) och klicka på knappen Install Selected.
5. Markera alternativet Virtual Devices och klicka på knappen New.
6. Ange följande inställningar:
   - Namn på den virtuella enheten
   - Mål-API, t.ex. Android 2.3, API-nivå 8
   - Storlek på SD-kortet (t.ex. 1 024)
   - Skal (t.ex. standard-HVGA)
7. Klicka på knappen Create AVD.

Det kan ta en stund att skapa den virtuella enheten beroende på systemets konfiguration.

Nu kan du starta den nya virtuella enheten.

1. Välj Virtual Device i programmet AVD Manager. Den virtuella enhet du skapade bör finnas med i listan.
2. Välj virtuell enhet och klicka på knappen Start.
3. Klicka på knappen Launch i nästa fönster.

Ett emulatorfönster bör nu öppnas på skrivbordet. Det kan ta några sekunder. Det kan också ta en liten stund för Android-operativsystemet att initieras. Du kan installera program som paketerats med målen *apk-debug* och *apk-emulator* på en emulator. Program som paketerats med målet *apk* fungerar inte på en emulator.

Fler hjälpavsnitt

http://developer.android.com/guide/developing/tools/othertools.html#android

http://developer.android.com/guide/developing/tools/emulator.html

Uppdatera AIR-program för mobiler

AIR-program för mobiler distribueras som systemspecifika paket och använder därför standardfunktionerna för uppdatering i andra program på plattformen. Vanligtvis innebär detta att programmet måste skickas till samma Market eller App Store som användes för att distribuera originalprogrammet.

AIR-program för mobiler kan inte använda AIR-klassen Updater eller uppdateringsramverket i AIR.

Uppdatera AIR-program på Android

Om ditt program distribueras via Android Market kan du uppdatera det genom att placera en ny version på Market, förutsatt att följande krav uppfylls (krävs av Market, inte av AIR):

- APK-paketet har signerats med samma certifikat.
- AIR-ID:t är samma.
- Värdet `versionNumber` i programbeskrivningen är större. (Du bör också öka värdet `versionLabel`, om det används.)

Användare som har hämtat ditt program från Android Market får veta att det finns en uppdatering via enhetens inbyggda programvara.

Fler hjälpavsnitt

För Android-utvecklare: Publicera uppdateringar på Android Market (på engelska)

Uppdatera AIR-program på iOS

Om ditt AIR-program distribueras via iTunes App Store kan du uppdatera det genom att skicka en uppdatering till App Store, förutsatt att följande krav uppfylls (krävs av Apple App Store, inte av AIR):

- Kodsigneringscertifikatet och provisioneringsprofilerna utfärdas till samma Apple-ID.
- IPA-paketet använder samma paket-ID från Apple.
- Uppdateringen minskar inte gruppen med enheter som stöds. Det innebär att om det ursprungliga programmet har stöd för enheter med iOS 3, kan du inte skapa en uppdatering som saknar stöd för iOS 3.

Viktigt! *Eftersom AIR SDK 2.6 och senare inte har stöd för iOS 3, medan AIR 2 har det, kan du inte uppdatera publicerade iOS-program som utvecklats med AIR 2 med en uppdatering som utvecklats med AIR 2.6+.*

Använda push-meddelanden

Med push-meddelanden kan fjärrmeddelandeleverantörer skicka meddelanden till program som körs på en mobil enhet. AIR 3.4 har stöd för push-meddelanden för iOS-enheter som använder Apple Push Notification-tjänsten (APN:er).

Obs! *Om du vill aktivera push-meddelanden för ett AIR for Android-program ska du använda ett ANE-tillägg, till exempel as3c2dm, utvecklad av Adobe-användaren Piotr Walczyszyn.*

I återstoden av detta avsnitt beskrivs hur du aktiverar push-meddelanden i AIR for iOS-program.

Obs! *I den här beskrivningen antas att du har ett ID som Apple-utvecklare, känner till arbetsflödet för iOS-utveckling och att du har skapat minst ett program på en iOS-enhet.*

Översikt över push-meddelanden

Apple Push Notification-tjänsten (APN:er) används av fjärrmeddelandeleverantörer för att skicka meddelanden till program som körs på iOS-enheter. En APN har stöd för följande meddelandetyper:

- Varningar
- Bagde-program
- Ljud

Fullständig information om APN:er finns på developer.apple.com.

Att använda push-meddelanden i program omfattar flera aspekter:

- **Klientprogram** – Register för push-meddelanden kommunicerar med fjärrmeddelandeleverantörer och erhåller push-meddelanden.
- **iOS** – Hanterar interaktionen mellan klientprogrammet och APN:er.
- **APN:er** – Tillhandahåller ett tokenID under klientregistreringen och skickar meddelanden från fjärrmeddelandeleverantören till iOS.
- **Fjärrmeddelandeleverantör** – Lagrar information för tokenId-klientprogrammet och skickar meddelanden till APN:er.

Arbetsflödet för registrering

Arbetsflödet för registrering av push-meddelanden med en tjänst på serversidan är följande:

1. Klientprogrammet begär att iOS aktiverar push-meddelanden.
2. iOS skickar denna begäran till APN:er.
3. APN-servern returnerar ett token-ID till iOS.
4. iOS returnerar detta token-ID till klientprogrammet.
5. Klientprogrammet (med hjälp av en programspecifik metod) levererar detta token-ID till fjärrmeddelandeleverantören, som lagrar detta för att använda det för push-meddelanden.

Arbetsflödet för meddelanden

Arbetsflödet för meddelanden är följande:

1. Fjärrmeddelandeleverantören skapar ett meddelande och skickar det till APN:er tillsammans med aktuellt token-ID.
2. APN:er vidarekopplar meddelanden till iOS på enheten.
3. iOS vidarekopplar meddelanden till programmet.

API för push-meddelanden

I AIR 3.4 introducerades en uppsättning API:er som har stöd för push-meddelanden. Dessa API:er finns i paketet `flash.notifications` och innehåller följande klasser:

- `NotificationStyle` – Definierar konstanter för meddelandentyperna: `ALERT`, `BADGE` och `SOUND`.C
- `RemoteNotifier` – Prenumerera och avbryta prenumeration på push-meddelanden.
- `RemoteNotifierSubscribeOptions` – Välja vilka meddelandetyper som ska erhållas. Använd egenskapen `notificationStyles` för att definiera en vektor med strängar som registrerar för flera meddelandetyper.

AIR 3.4 innehåller även `flash.events.RemoteNotificationEvent`, som skickas av `RemoteNotifier` enligt följande:

- När en programprenumeration skapas och ett nytt token-ID erhålls från APN:er.
- När ett nytt fjärrmeddelande tas emot.

Dessutom skickar `RemoteNotifier` en `flash.events.StatusEvent` om ett fel upptäcks under prenumerationsprocessen.

Hantera push-meddelanden i ett program

Gör så här för att registrera programmet för push-meddelanden:

- Skapa kod som prenumererar på push-meddelanden i programmet.
- Aktivera push-meddelanden i programmets XML-fil.
- Skapa en provisioneringsprofil och ett certifikat som aktiverar push-tjänster för iOS.

I följande exempel visas kod för hur du prenumererar på push-meddelanden och hanterar push-meddelandehändelser:

```
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());
}
}
}
```

Aktivera push-meddelanden i programmets XML-fil

Ange följande i taggen `Entitlements` (under taggen `iphone`) för att kunna använda push-meddelanden i programmet:

```
<iphone>
                    ...
                       <Entitlements>
                          <![CDATA[
                             <key>aps-environment</key>
                             <string>development</string>
                          ]]>
                       </Entitlements>
                    </iphone>
```

När du ska skicka programmet till App Store, ändrar du ett `<string>`-element för utveckling till produktion:

```
<string>production</string>
```

Om programmet har stöd för översatta strängar, anger du språket i taggen `supportedLanguages` under taggen `intialWindow` enligt exemplet nedan:

```
<supportedLanguages>en de cs es fr it ja ko nl pl pt</supportedLanguages>
```

Skapa en provisioneringsprofil och ett certifikat som aktivera push-tjänster för iOS

Du aktiverar kommunikation för program-APN:er genom att paketera programmet med en provisioneringsprofil och ett certifikat som aktiverar push-tjänster för iOS enligt följande:

1. Logga in på ditt konto för Apple-utvecklare.
2. Navigera till Provisioning Portal.
3. Klicka på fliken Program-ID (App IDs).
4. Klicka på knappen Nytt program-ID (New App ID).
5. Ange en beskrivning och källidentifierare (du ska inte använda * i källidentifieraren).
6. Klicka på Skicka (Submit). På Provisioning Portal skapas ditt program-ID och sidan för program-ID:n visas på nytt.

7 Klicka på Konfigurera (Configure) (till höger om ditt program-ID). Sidan Konfigurera program-ID (Configure App ID) öppnas.

8 Markera kryssrutan Aktivera för APN-tjänst (Enable for Apple Push Notification service). Observera att det finns två typer av push SSL-certifikat: ett för utveckling/testning och ett för produktion.

9 Klicka på knappen Konfigurera (Configure) till höger om Utveckla push-SSL-certifikat (Development Push SSL Certificate). CSR-sidan (Generate Certificate Signing Request) öppnas.

10 Skapa ett CSR med hjälp av verktyget Keychain Access enligt instruktionerna på sidan.

11 Skapa SSL-certifikatet.

12 Hämta och installera SSL-certifikatet.

13 (Valfritt) Upprepa stegen 9 till 12 för produktions-push-SSL-certifikatet.

14 Klicka på Klar (Done). Sidan Konfigurera program-ID (Configure App ID) öppnas.

15 Klicka på Klar (Done). Sidan Program-ID:n (App IDs) öppnas. Lägg märke till den gröna cirkeln intill push-meddelandet för ditt program-ID.

16 Se till att spara SSL-certifikaten eftersom de används senare för program- och leverantörskommunikation.

17 Klicka på fliken Provisionering (Provisioning) för att öppna sidan Provisioneringsprofiler (Provisioning Profiles).

18 Skapa en provisioneringsprofil för ditt nya program-ID och hämta det.

19 Klicka på fliken Certifikat (Certificates) och hämta ett nytt certifikat för den nya provisioneringsprofilen.

Använda ljud för push-meddelanden

Om du vill aktivera ljudmeddelanden för ditt program paketerar du ljudfilen på samma sätt som du gör med andra resurser, men i samma katalog som SWF- och app-xml-filerna. Till exempel:

```
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 har stöd för följande ljuddataformat (i aiff-, wav- eller caf-filer):

- Linear PCM
- MA4 (IMA/ADPCM)
- uLaw
- aLaw

Använda översatta varningsmeddelanden

Om du vill använda översatta varningsmeddelanden i ditt program ska du paketera dina översatta strängar som lproj-mappar. Gör så här för att skapa varningsmeddelanden på spanska:

1 Skapa en es.lproj-mapp på samma nivå som app-xml-filen.

2 Skapa i es.lproj-mappen en textfil med namnet Localizable.Strings.

3 Öppna filen Localizable.Strings i ett textredigeringsprogram och lägg till meddelandenycklar och motsvarande översatta strängar. Till exempel:

```
"PokeMessageFormat" = "La notificación de alertas en español."
```

4 Spara filen.

5 När programmet erhåller ett varningsmeddelande med detta nyckelvärde och enhetsspråket är spanska kommer det översatta textmeddelandet att visas.

Konfigurera en fjärrmeddelandeleverantör

Du behöver en fjärrmeddelandeleverantör för att kunna skicka push-meddelanden till ditt program. Detta serverprogram fungerar som en leverantör som accepterar push-indata och som skickar meddelanden och meddelandedata till APN:er, som i sin tur skickar push-meddelanden till ett klientprogram.

Mer information om push-meddelanden från en fjärrmeddelandeleverantör finns i Provider Communication with Apple Push Notification Service i Apple Developer Library.

Alternativ för fjärrmeddelandeleverantörer

Följande alternativ finns för en fjärrmeddelandeleverantör:

- Skapa en egen leverantör baserad på APNS-php öppen källserver. Du ställer in en PHP-server med hjälp av http://code.google.com/p/apns-php/. Med detta Google Code-projekt kan du skapa ett gränssnitt som matchar dina specifika krav.
- Använd en tjänsteleverantör. På http://urbanairship.com/ finns ett exempel på en färdig APN-leverantör. Sedan du registrerat tjänsten fortsätter du med att tillhandahålla din enhets-token med hjälp av en kod enligt följande:

```
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);
                                    }
```

Du kan sedan skicka testmeddelanden med verktyget Urban Airship.

Certifikat för fjärrmeddelandeleverantören

Du måste kopiera SSL-certifikatet och den privata nyckeln (skapad tidigare) till rätt plats på fjärrmeddelandeleverantörens server. Du kombinerar vanligtvis dessa båda filer till en `.pem`-fil. Gör så här:

1. Öppna ett terminalfönster.
2. Skapa en `.pem`-fil från SSL-certifikatet genom att skriva följande kommando:

   ```
   openssl x509 -in aps_developer_identity.cer -inform der -out TestPushDev.pem
   ```

3. Skapa en .pem-fil av den privata nyckelfilen (`.p12`) genom att skriva följande kommando:

   ```
   openssl pkcs12 -nocerts -out TestPushPrivateKey.pem -in certificates.p12
   ```

4. Kombinera de båda .pem-filerna till en fil genom att skriva följande kommando:

   ```
   cat TestPushDev.pem TestPushPrivateKey.pem > FinalTestPush.pem
   ```

5. Lägg till den kombinerade `.pem`-filen hos serverleverantören när du skapar push-programmet på serversidan.

Mer information finns på Installing the SSL Certificate and Key on the Server i Apple Local and Push Notification Programming Guide.

Administrera push-meddelanden i ett program

Så här administrerar du push-meddelanden i ett program:

- Global användarkonfiguration och godkännande av push-meddelanden
- Få användargodkännande av enskilda push-meddeladen
- Hantera push-meddelanden och meddelandedata

Konfigurera och godkänna push-meddelanden

Första gången en användare startar ett program aktiverat för push-meddelanden, visas i iOS en dialogruta ***appname* Vill du skicka push-meddelanden (Would Like to Send You Push Notifications)** med knapparna Tillåt inte (Don't Allow) och OK. Om användaren väljer OK kan programmet ta emot alla stilar för meddelanden som omfattas av prenumerationen. Om du väljer Tillåt inte (Don't Allow) tas inga meddelanden emot.

Obs! *Användarna kan även välja Inställningar (Settings) > Meddelanden (Notifications) för att kontrollera specifika meddelandetyper som de kan erhålla för varje push-aktiverat program.*

Apple rekommenderar att varje gång som ett program aktiveras ska det prenumerera på push-meddeladen. När programmet anropar `RemoteNotifier.subscribe()`, erhålls en `RemoteNotificationEvent`-händelse av typen `token`, som innehåller ett 32-byte stort unikt numeriskt token-ID som unikt identifierar programmet på enheten.

När enheten erhåller ett push-meddelande visas ett poup-fönster med knapparna Stäng (Close) och Starta (Launch). Om användarna pekar på Stäng (Close) händer inget, men om användaren pekar på Starta (Launch), öppnar iOS programmet och det erhåller en `flash.events.RemoteNotificationEvent`-händelse av typen `notification`. Se nedan.

Hantera push-meddelanden och meddelandedata

När fjärrmeddelandeleverantören skickar ett meddelande till en enhet (med aktuellt tokenID), kommer programmet att erhålla händelsen `flash.events.RemoteNotificationEvent` med typen `notification`, oberoende av om programmet körs eller inte. Vid den här tidpunkten utför programmet en programspecifik meddelandebearbetning. Om programmet hanterar meddelande data öppnar du det genom den JSON-formaterade egenskapen `RemoteNotificationEvent.data`.

Kapitel 8: Utveckla AIR-program för tv-enheter

AIR-funktioner för tv-enheter

Du kan skapa Adobe® AIR®-program för tv-enheter, som tv-apparater, digitala videoinspelare och Blu-ray-skivspelare, om enheten har Adobe AIR for TV. AIR for TV är optimerat för tv-enheter och utnyttjar till exempel en enhets maskinvaruacceleration för video och bilder med höga prestanda.

AIR-program för tv-enheter är SWF-baserade program, inte HTML-baserade. AIR for TV-programmet kan utnyttja maskinvaruacceleration såväl som andra AIR-funktioner, som passar bra för ”vardagsrumsmiljön”.

Enhetsprofiler

I AIR används profiler för att definiera olika målenheter med liknande funktioner. Använd följande profiler för AIR for TV-program:

- Profilen `tv`. Använd den här profilen i AIR-program som har en AIR for TV-enhet som målenhet.
- Profilen `extendedTV`. Använd den här profilen om AIR for TV-programmet använder ANE-tillägg.

De ActionScript-funktioner som definierats för de här profilerna beskrivs i ”Enhetsprofiler” på sidan 241. Särskilda ActionScript-skillnader för AIR for TV-program behandlas i Referenshandbok för ActionScript 3.0 i Adobe Flash-plattformen.

Mer information om AIR for TV-profiler finns i ”Profiler som stöds” på sidan 137.

Maskinvaruacceleration

Tv-enheter har maskinvaruacceleration som dramatiskt ökar grafik- och videoprestanda i AIR-programmet. Läs avsnittet ”Om utformning av AIR for TV-program” på sidan 120 om du vill veta hur du utnyttjar maskinvaruacceleration.

Innehållsskydd

Med AIR for TV kan du skapa nyanserade och mångfasetterade konsumentprodukter runt videoinnehåll, oavsett om det gäller stora Hollywood-produktioner, independentfilmer eller tv-program. Innehållsleverantörer kan skapa interaktiva program med hjälp av Adobes verktyg. De kan integrera Adobes serverprodukter i deras infrastruktur för innehållsdistribution eller arbeta med någon av Adobes ecosystem-partners.

Innehållsskydd är ett nyckelkrav för distribution av betalvideofilm. AIR for TV har stöd för Adobe® Flash® Access™, en lösning för innehållsskydd och betaltjänster som uppfyller de strikta säkerhetskrav som ställs av innehållsägarna (inklusive de stora filmbolagen).

Flash Access har stöd för följande:

- Direktuppspelning och hämtning av video.
- Olika affärsmodeller, inklusive annonsbaserade prenumerationer, uthyrning och elektronisk försäljning (EST).
- Olika system för innehållsleverans, inklusive dynamisk HTTP-direktuppspelning, direktuppspelning via RTMP (Real Time Media Protocol) med Flash® Media Server och progressiv nedladdning med HTTP.

AIR for TV har även inbyggt stöd för RTMPE, den krypterade versionen av RTMP, för befintliga direktuppspelningslösningar med lägre säkerhetskrav. RTMPE och relaterade tekniker för SWF-verifiering stöds i Flash Media Server.

Du hittar mer information i Adobe Flash Access.

Flerkanalsljud

Från och med AIR 3 har AIR for TV stöd för flerkanalsljud för videofilmer som fortlöpande laddas ned från en HTTP-server. Detta stöd omfattar följande kodekar:

- AC-3 (Dolby Digital)
- E-AC-3 (Enhanced Dolby Digital)
- DTS Digital Surround
- DTS Express
- DTS-HD High Resolution Audio
- DTS-HD Master Audio

Obs! *Stöd för flerkanalsljud i videofilmsflöden från en FMS-server (Adobe Flash Media Server) är inte tillgängligt för tillfället.*

Indata från spel

Från och med AIR 3 har AIR for TV stöd för ActionScript API:er som tillåter att program kan kommunicera med spelindataenheter såsom styrspakar, rattar och trollstavar. Trots att dessa enheter kallas spelindataenheter är det inte bara i AIR for TV-spel som dessa enheter kan användas.

Det finns ett stort antal olika spelindataenheter med olika kapacitet tillgängliga. Detta innebär att enheterna är API-generaliserade så att ett program kan fungera bra med olika (och möjligtvis okända) typer av spelindataenheter.

Klassen GameInput är startpunkten för ActionScript API:ernas spelindata. Mer information finns på GameInput.

3D-accelererad grafikåtergivning för scenen

Från och med AIR 3 har AIR for TV stöd för 3D-accelererad grafikåtergivning för scenen. ActionScript API:erna för Stage3D är en uppsättning av GPU-accelererade API:er på lågnivå för aktivering av avancerade 2D- och 3D-funktioner. Dessa lågnivå-API:er ger utvecklarna flexibilitet att anpassa GPU-maskinvaruacceleration för att få avsevärda prestandavinster. Du kan även använda spelmotorer som har stöd för ActionScript-API:er för Stage3D.

Mer information finns på Gaming engines, 3D, and Stage 3D.

ANE-tillägg

När programmet har `extendedTV`-profilen som mål kan du använda ANE-paket (AIR Native Extension).

Enhetstillverkare tillhandahåller vanligtvis ANE-paket för att ge tillgång till enhetsfunktioner som annars inte stöds av AIR. Ett ANE-tillägg kan till exempel ge dig möjlighet att byta kanal på en tv eller pausa uppspelningen på en videospelare.

När du paketerar ett AIR for TV-program, som använder ANE-paket, paketerar du programmet i en AIRN-fil i stället för i en AIR-fil.

ANE-tillägg för AIR for TV-enheter är alltid *enhetsanknutna*. Detta betyder att ANE-bibliotek installeras på AIR for TV-enheten. Det ANE-paket som du inkluderar i programpaketet innehåller *aldrig* tilläggets ANE-bibliotek. Ibland innehåller det en ActionScript-anpassad version av ANE-tillägget. Den här ActionScript-versionen är en stub eller simulator till tillägget. Det är enhetstillverkaren som installerar det äkta tillägget, inklusive ANE-bibliotek, på enheten.

Om du utvecklar ANE-tillägg ska du tänka på följande:

- Tillfråga alltid enhetstillverkaren om du skapar ett ANE-tillägg för AIR for TV för deras enhet.
- För vissa AIR for TV-enheter är det endast enhetstillverkaren som skapar enhetstillägg.
- För alla AIR for TV-enheter gäller att det är enhetstillverkaren som bestämmer vilka ANE-tillägg som ska installeras.
- Utvecklingsverktygen för ANE-tilläggen för AIR for TV varierar mellan olika tillverkare.

Mer information om hur du använder ANE-tillägg i AIR-program finns i ”Använda ANE-tillägg för Adobe AIR” på sidan 145.

Mer information hur du skapar ANE-tillägg finns i Utveckla ANE-tillägg för Adobe AIR.

Om utformning av AIR for TV-program

Att tänka på vid videoanvändning

Riktlinjer för videokodning

Vid direktuppspelning av video till en tv-enhet bör du följa dessa kodningsriktlinjer:

Videokodek:	H.264, vanlig eller hög profil, progressiv kodning
Upplösning:	720i, 720p, 1080i eller 1080p
Bildrutefrekvens:	24 bildrutor per sekund eller 30 bildrutor per sekund
Ljudkodek	AAC-LC eller AC-3, 44,1 kHz, stereo, eller dessa flerkanalsljudkodekar: E-AC-3, DTS, DTS Express, DTS-HD High Resolution Audio eller DTS-HD Master Audio
Kombinerad bithastighet:	upp till 8 Mbit/s beroende på tillgänglig bandbredd
Ljudbithastighet:	upp till 192 kbit/s
Pixelproportioner:	1 × 1

Du bör använda H.264-kodeken för video som levereras till AIR for TV-enheter.

Obs! *AIR for TV har också stöd för video som kodats med Sorenson Spark- eller On2 VP6-kodekar. Maskinvaran kan däremot inte avkoda och visa dessa kodekar. I stället avkodas och visas dessa kodekar i körningsmiljön med hjälp av programvara, och därför spelas videon upp med en betydligt lägre bildrutefrekvens. Av den anledningen bör du använda H.264 om det går.*

Klassen StageVideo

AIR for TV har stöd för maskinvaruavkodning och visning av H.264-kodad video. Använd klassen StageVideo för att aktivera den här funktionen.

Avsnittet Använda klassen StageVideo för maskinvaruaccelererad presentation i *Utvecklarhandbok för ActionScript 3.0* innehåller information om:

- API:t för klassen StageVideo och relaterade klasser.
- begränsningar för klassen StageVideo.

För att ge bästa möjliga stöd till befintliga AIR-program som använder Video-objektet för H.264-kodad video, används ett StageVideo-objekt *internt* i AIR for TV. Det innebär att videouppspelningen kan utnyttja fördelarna med maskinvaruavkodning och visning. Dock har Video-objektet samma begränsningar som ett StageVideo-objekt. Om programmet till exempel försöker rotera videon händer ingenting, eftersom maskinvaran, inte miljön, visar videon.

När du skriver nya program kan du däremot använda StageVideo-objektet för H.264-kodad video.

Du hittar ett exempel på användning av StageVideo-klassen i Leverera video och innehåll för Flash-plattformen på tv.

Riktlinjer för videoleverans

På en AIR for TV-enhet kan nätverkets tillgängliga bandbredd variera under videouppspelning. De här variationerna kan till exempel inträffa när en annan användare börjar använda samma Internet-anslutning.

Du bör därför se till att ditt system för videoleverans använder funktioner för variabel bithastighet. På serversidan har till exempel Flash Media Server stöd för variabel bithastighet. På klientsidan kan du använda Open Source Media Framework (OSMF).

Följande protokoll är tillgängliga för leverans av videoinnehåll via nätverket till ett AIR for TV-program:

- Dynamisk HTTP- och HTTPS-direktuppspelning (F4F-format)
- RTMP-, RTMPE-, RTMFP-, RTMPT- och RTMPTE-direktuppspelning
- Progressiv nedladdning för HTTP och HTTPS

Mer information finns i följande avsnitt:

- Adobe Flash Media Server Developer's Guide
- Open Source Media Framework

Att tänka på vid ljudanvändning

ActionScript-koden för att spela upp ljud är likadan i AIR for TV-program som i andra AIR-program. Du hittar mer information i Arbeta med ljud i *Utvecklarhandbok för ActionScript 3.0*.

När det gäller flerkanalsljud i AIR for TV ska du tänka på följande:

- AIR for TV har stöd för flerkanalsljud för videofilmer som progressivt laddas ned från en HTTP-server. Stöd för flerkanalsljud i videofilmsflöden från en FMS-server (Adobe Flash Media Server) är inte tillgängligt för tillfället.
- Trots att AIR for TV har stöd för många ljudkodekar, är det inte alla AIR for TV-*enheter* som har stöd för samtliga. Använd flash.system.Capabilities-metoden `hasMultiChannelAudio()` för att kontrollera om en AIR for TV-enhet har stöd för en speciell ljudkodek för flera kanaler, till exempel AC-3.

  Tänk dig ett program som fortlöpande hämtar en videofil från en server. Servern har olika H.264-videofiler som har stöd för olika flerkanalsljudkodekar. Programmet kan använda `hasMultiChannelAudio()` för att bestämma vilken videofil som ska begäras från servern. Programmet kan även skicka strängen i `Capabilities.serverString` till servern. Strängen anger vilken flerkanalsljudkodek som är tillgänglig vilket gör att servern kan välja rätt videofil.
- När en av DTS-ljudkodekarna används finns fall där `hasMultiChannelAudio()` returnerar `true` trots att DTS-ljudet inte spelas upp.

Tänk dig till exempel att du har en Blu-ray-spelare, med en S/PDIF-utgång, ansluten till en gammal förstärkare. Den gamla förstärkaren saknar DTS-stöd, men för S/PDIF finns inget protokoll för att meddela Blu-ray-spelaren. Om Blu-ray-spelaren skickar DTS-strömmen till den gamla förstärkaren kommer användaren inte att höra något. Det är därför bäst när DTS användas att tillhandhålla ett gränssnitt så att användaren kan upptäcka om inget ljud spelas upp. Därefter kan programmet återställas till en annan kodek.

I följande tabell visas när olika ljudkodekar ska användas i AIR for TV-program. Av tabellen framgår även när AIR for TV-enheter använder maskinvaruacceleratorer för att avkoda en ljudkodek. Maskinvaruavkodning förbättrar prestandan och avlastar processorn.

Ljudkodek	**Tillgänglighet på AIR for TV-enhet**	**Maskinvaruavkodning**	**Använd denna videokodek...**	**Mer information**
AAC	Alltid	Alltid	För video som kodats med H.264. För direktuppspelning av ljud, t.ex. via en Internet-musiktjänst.	Om du använder en AAC-ström med bara ljud kapslar du in ljudströmmen i en MP4-behållare.
mp3	Alltid	Nej	För ljud i programmets SWF-filer. I videofilmer som avkodas med Sorenson Spark eller On2 VP6.	Ljudet spelas inte upp i AIR for TV-enheter när mp3 används i en H.264-video.
AC-3 (Dolby Digital) E-AC-3 (Enhanced Dolby Digital) DTS Digital Surround DTS Express DTS-HD High Resolution Audio DTS-HD Master Audio	Kontrollera	Ja	För video som kodats med H.264.	Det vanliga är att AIR for TV skickar en flerkanalsljudström till en extern ljud-/videomottagare som avkodar och spelar upp ljudet.
Speex	Alltid	Nej	Tar emot ett direktuppspelande röstflöde.	Ljudet spelas inte upp i AIR for TV-enheter när Speex används i en H.264-video. Använd Speex endast för videofilmer som är kodade med Sorenson Spark eller On2 VP6.
NellyMoser	Alltid	Nej	Tar emot ett direktuppspelande röstflöde.	Ljudet spelas inte upp i AIR for TV-enheter när NellyMoser används i en H.264-video. Använd NellyMoser endast för videofilmer som är kodade med Sorenson Spark eller On2 VP6.

Obs! *Vissa videofiler innehåller två ljudströmmar. En videofil kan till exempel innehålla både en AAC- och en AC3-ström. I AIR for TV finns inte stöd för sådana videofiler och om du använder sådana filer kan resultatet bli att inget ljud hörs från videon.*

Maskinvaruacceleration för grafik

Använda maskinvaruacceleration för grafik

AIR for TV-enheter har maskinvaruacceleration för 2D-grafikåtgärder. Enhetens maskinvaruacceleratorer för grafik avlastar huvudprocessorn och hanterar följande åtgärder:

- Bitmappsåtergivning

- Bitmappsskalning
- Bitmappsblandning
- Fyllning av enfärgade rektanglar

Den här maskinvaruaccelerationen innebär att många grafikåtgärder i AIR for TV-program kan få höga prestanda. Bland de här åtgärderna finns:

- Glidande övergångar
- Skalande övergångar
- Tona in och ut
- Sammansättning av flera bilder med alfa

Använd någon av följande tekniker för att utnyttja prestandafördelarna med maskinvaruacceleration för grafik för den här typen av åtgärder:

- Ange egenskapen `cacheAsBitmap` som `true` för MovieClip-objekt och andra visningsobjekt med innehåll som till större delen är oföränderligt. Utför sedan glidande övergångar, tonande övergångar och alfablandning på de här objekten.
- Använd egenskapen `cacheAsBitmapMatrix` på visningsobjekt som du vill skala eller översätta (använd x- och y-omplacering).

  När åtgärder i Matrix-klassen används för skalning och översättning utförs åtgärderna av enhetens maskinvaruacceleratorer. Ett annat scenario är när du ändrar dimensionerna på ett visningsobjekt, som har egenskapen `cacheAsBitmap` inställd på `true`. När dimensionerna ändras ritas bitmappen om av miljöns programvara. Omritning med programvara medför sämre prestanda än skalning med maskinvaruacceleration via en Matrix-åtgärd.

  Tänk dig till exempel ett program som visar en bild, som expanderar när slutanvändaren markerar den. Använd Matrix-skalningsåtgärden flera gånger för att ge intryck av att bilden expanderas. Men beroende på storleken på originalbilden och den slutliga bilden kan kvaliteten på den slutliga bilden bli oacceptabel. Därför återställer du dimensionerna på visningsobjektet efter att expanderingsåtgärderna har slutförts. Eftersom `cacheAsBitmap` är `true` ritas visningsobjektet om av körningsmiljön, men bara en gång, och en bild med hög kvalitet återges.

  Obs! *AIR for TV-enheter har vanligtvis inte stöd för maskinvaruaccelererad rotation och skevning. Om du anger rotation och skevning i klassen Matrix utförs därför alla Matrix-åtgärder i programvaran. De här programvaruåtgärderna kan ha en negativ effekt på prestanda.*
- Använd klassen BitmapData för att skapa ett eget bitmappscachningbeteende.

Här hittar du mer information om bitmappscachning:

- Cachelagra visningsobjekt
- Bitmappscachning
- Manuell bitmappscachning

Hantera grafikminne

För att utföra accelererade grafikåtgärder använder maskinvaruacceleratorer ett särskilt grafikminne. Om ditt program använder hela grafikminnet går programmet långsammare, eftersom AIR for TV återgår till att använda programvara för grafikåtgärder.

Så här hanterar du programmets användning av grafikminne:

- När du är klar med användningen av en bild eller andra bitmappdata frigör du dess associerade grafikminne. Det gör du genom att anropa metoden `dispose()` för Bitmap-objektets `bitmapData`-egenskap. Till exempel:

  ```
  myBitmap.bitmapData.dispose();
  ```

  Obs! *När referensen till BitmapData-objektet frigörs blir grafikminnet inte omedelbart ledigt. Miljöns skräpinsamlare frigör så småningom grafikminnet, men med ett anrop till* `dispose()` *får ditt program större kontroll.*

- Använd PerfMaster Deluxe, ett AIR-program från Adobe, för att få en bättre förståelse för maskinvaruacceleration för grafik på målenheten. Det här programmet visar antalet bildrutor per sekund för att köra olika åtgärder. Använd PerfMaster Deluxe för att jämföra olika implementeringar av en och samma åtgärd. Du kan till exempel jämföra flyttning av en bitmappsbild med flyttning av en vektorbild. Du kan hämta PerfMaster Deluxe på Flash-plattform för TV.

Hantera visningslistan

Om du vill göra ett visningsobjekt synligt ska du ställa in `false` för objektets `visible`-egenskap. Det kommer att finnas kvar i visningslistan, men AIR for TV kommer inte att återge eller visa det. Den här tekniken är användbar för objekt som ofta visas och tas bort, eftersom den har en liten inverkan på bearbetningsbelastningen. Om emellertid egenskapen `visible` sätts till `false` kommer inga av objektets resurser att frisläppas. Därför ska du, när objektet visats klart eller om det inte ska visas på en längre tid, ta bort det från visningslistan. Dessutom ska alla referenser till objektet sättas till `null`. Dessa åtgärder gör att skräpsamlaren kan frigöra objektets resurser.

Använda PNG- och JPEG-bilder

Två vanliga bildformat i program är PNG och JPEG. När det gäller dessa bildformat i AIR for TV-program bör du tänka på följande:

- AIR for TV använder vanligen maskinvaruacceleration för att avkoda JPEG-filer.
- AIR for TV använder vanligen programvara för att avkoda PNG-filer. Att avkoda PNG-filer i programvara går fort.
- PNG är det enda bitmappsformatet för flera plattformar som stödjer genomskinlighet (som alfakanal).

Därför bör du använda bildformaten i dina program enligt följande:

- Använd JPEG-filer för fotografier för att utnyttja maskinvaruaccelererad avkodning.
- Använd PNG-filer för element i användargränssnittet. Element i användargränssnittet kan ha alfainställningar, och programvaruavkodning ger tillräckligt snabba prestanda för dessa.

Scenen i AIR for TV-program

När du skapar ett AIR for TV-program och arbetar med klassen Stage bör du tänka på följande:

- Skärmupplösning
- Säker yta för visning
- Scenens skalningsläge
- Scenens justering
- Scenens visningsläge
- Design för flera skärmstorlekar
- Scenens kvalitetsinställning

Skärmupplösning

För tillfället har tv-enheter oftast en av dessa upplösningar: 540p, 720p eller 1080p. De här skärmupplösningarna resulterar i följande värden i ActionScript-klassen Capabilities:

Skärmupplösning	`Capabilities.screenResolutionX`	`Capabilities.screenResolutionY`
540p	960	540
720p	1280	720
1080p	1920	1 080

Om du vill skriva en helskärmsversion av ett AIR for TV-program för en viss enhet hårdkodar du `Stage.stageWidth` och `Stage.stageHeight` efter enhetens skärmupplösning. Om du däremot vill skriva en helskärmsversion av ett program som ska köras på flera enheter använder du egenskaperna `Capabilities.screenResolutionX` och `Capabilities.screenResolutionY` för att ange scenens dimensioner.

Till exempel:

```
stage.stageWidth = Capabilities.screenResolutionX;
stage.stageHeight = Capabilities.screenResolutionY;
```

Säker yta för visning

Den *säkra ytan för visning* på en tv är ett område på skärmen som är lite indraget från skärmens kanter. Det här området är så pass indraget att slutanvändaren kan se hela området, utan att tevens kantlist skymmer någon del av det. Eftersom den här kantlisten inte ser likadan ut hos alla tillverkare varierar det indrag som krävs också. Med hjälp av den säkra ytan för visning blir det enklare att avgöra vilken del av skärmen som syns. Den säkra ytan för visning kallas även *titelsäker yta*.

Överskanning avser den del av skärmen som inte är synlig för att den döljs av kantlisten.

Du bör använda ett indrag på 7,5 % på vardera sida av skärmen. Till exempel:

Den säkra ytan för visning för en skärmupplösning på 1 920 x 1 080

Ha alltid den säkra ytan för visning i åtanke när du skapar en helskärmsversion av ett AIR for TV-program:

- Använd hela skärmen för bakgrunder, till exempel bakgrundsbilder eller bakgrundsfärger.
- Använd bara den säkra ytan för visning för viktiga programelement som text, grafik, video och objekt i användargränssnittet (som knappar).

Följande tabell visar dimensionerna på den säkra ytan för visning för var och en av skärmupplösningarna, med ett indrag på 7,5 %.

Skärmupplösning	Bredd och höjd på den säkra ytan för visning	Bredd på vänster och höger indrag	Bredd på övre och nedre indrag
960 x 540	816 x 460	72	40
1 280 x 720	1 088 x 612	96	54
1 920 x 1 080	1 632 x 918	144	81

Det allra bästa är emellertid att alltid beräkna den säkra ytan för visning dynamiskt. Till exempel:

```
var horizontalInset, verticalInset, safeAreaWidth, safeAreaHeight:int;

horizontalInset = .075 * Capabilities.screenResolutionX;
verticalInset = .075 * Capabilities.screenResolutionY;
safeAreaWidth = Capabilities.screenResolutionX - (2 * horizontalInset);
safeAreaHeight = Capabilities.screenResolutionY - (2 * verticalInset);
```

Scenens skalningsläge

Ange `Stage.scaleMode` som `StageScaleMode.NO_SCALE` och lyssna efter resize-händelser för scenen.

```
stage.scaleMode = StageScaleMode.NO_SCALE;
stage.addEventListener(Event.RESIZE, layoutHandler);
```

Med den här inställningen blir scenens koordinater samma som pixelkoordinaterna. Tillsammans med visningsläget `FULL_SCREEN_INTERACTIVE` och scenjusteringen `TOP_LEFT` kan du med den här inställningen använda den säkra ytan för visning effektivt.

I helskärmsprogram innebär det här skalningsläget att egenskaperna `stageWidth` och `stageHeight` för klassen Stage motsvarar egenskaperna `screenResolutionX` och `screenResolutionY` för klassen Capabilities.

Dessutom behåller scenens innehåll sin definierade storlek även om programmets fönsterstorlek ändras. Ingen automatisk layout eller skalning utförs av körningsmiljön. Dessutom skickas även en `resize`-händelse för klassen Stage när fönsterstorleken ändras. Det betyder att du har full kontroll över hur du anpassar programmets innehåll när programmet startas och när programfönstrets storlek ändras.

***Obs!** Beteendet `NO_SCALE` är samma som i alla AIR-program. I AIR for TV-program är användningen av den här inställningen avgörande för användningen av den säkra ytan för visning.*

Scenens justering

Ange `Stage.align` som `StageAlign.TOP_LEFT`:

```
stage.align = StageAlign.TOP_LEFT;
```

Den här justeringen placerar `0,0`-koordinaten i det övre vänstra hörnet av skärmen, vilket är praktiskt för innehållsplacering med ActionScript.

Tillsammans med skalningsläget `NO_SCALE` och visningsläget `FULL_SCREEN_INTERACTIVE` kan du med den här inställningen använda den säkra ytan för visning effektivt.

Scenens visningsläge

Ange `Stage.displayState` i en helskärmsversion av ett AIR for TV-program som `StageDisplayState.FULL_SCREEN_INTERACTIVE`:

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

Det här värdet anger att AIR-programmet ska expandera scenen till hela skärmen, med användarinmatning tillåten.

Du bör använda inställningen `FULL_SCREEN_INTERACTIVE`. Tillsammans med skalningsläget `NO_SCALE` och scenjusteringen `TOP_LEFT` kan du med den här inställningen använda den säkra ytan för visning effektivt.

För helskärmsprogram gör du därför följande i en hanterare för `ADDED_TO_STAGE`-händelsen i huvuddokumentklassen:

```
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;
}
```

Gör sedan följande i hanteraren för `RESIZE`-händelsen:

- Jämför skärmupplösningarna med scenens bredd och höjd. Om de är samma inträffade `RESIZE`-händelsen på grund av att scenens visningsläge ändrades till `FULL_SCREEN_INTERACTIVE`.
- Beräkna och spara dimensionerna för den säkra ytan för visning och motsvarande indrag.

```
private function onResize(evt:Event):void
{
    if ((Capabilities.screenResolutionX == stage.stageWidth) &&
        (Capabilities.screenResolutionY == stage.stageHeight))
    {

        // Calculate and save safe viewing area dimensions.
    }
}
```

När scenens mått är lika med `Capabilities.screenResolutionX` och `screenResolutionY`, kommer AIR for TV att få maskinvaran att ge bästa tänkbara video- och grafikåtergivning.

Obs! *Hur naturtroget grafik och video visas på en TV-skärm kan variera beroende på `Capabilities.screenResolutionX`- och `screenResolutionY`-värdena. Detta har att göra med på vilken enhet som AIR for TV körs. Till exempel om AIR for TV körs på en digitalbox med en upplösning på 1280 x 720 så kan den anslutna TV:n ha en skärmupplösning på 1920 x 1080. AIR for TV fungerar emellertid så att maskinvaran ger den bästa tänkbara återgivningen. I exemplet här visar maskinvaran en 1080p-video med en skärmupplösning på 1920 x 1080.*

Design för flera skärmstorlekar

Du kan utveckla samma AIR for TV-program (i helskärmsversion) så att det fungerar och ser bra ut på flera AIR for TV-enheter. Gör så här:

1. Ange egenskaperna `scaleMode`, `align` och `displayState` för scenen med de rekommenderade värdena: `StageScaleMode.NO_SCALE`, `StageAlign.TOP_LEFT` och `StageDisplayState.FULL_SCREEN_INTERACTIVE`.
2. Skapa den säkra ytan för visning baserat på `Capabilities.screenResolutionX` och `Capabilities.screenResolutionY`.
3. Anpassa storleken och layouten på ditt innehåll efter bredden och höjden på den säkra ytan för visning.

Även om objekten i ditt innehåll är stora, särskilt jämfört med mobilprogram, är begrepp som dynamisk layout, relativ placering och anpassningsbart innehåll samma. Mer information om ActionScript som stöd för dessa begrepp finns i Skapa Flash-innehåll för mobiler och flera skärmstorlekar.

Scenens kvalitet

Egenskapen `Stage.quality` för ett AIR for TV-program är alltid `StageQuality.High`. Du kan inte ändra den.

Den här egenskapen anger återgivningskvaliteten för alla Stage-objekt.

Indatahantering via fjärrkontroll

Användare använder oftast ditt AIR for TV-program via en fjärrkontroll. Du bör emellertid hantera dessa indata på samma sätt som du hanterar indata från ett tangentbord till en stationär dator. Du bör särskilt tänka på att hantera händelsen `KeyboardEvent.KEY_DOWN`. Du hittar mer information i Samla in tangentbordsindata i *Utvecklarhandbok för ActionScript 3.0*.

Knapparna på fjärrkontrollen mappar till ActionScript-konstanter. Styrknapparna på en fjärrkontroll mappas till exempel enligt följande:

Styrknapp på fjärrkontrollen	ActionScript 3.0-konstant
Upp	`Keyboard.UP`
Ned	`Keyboard.DOWN`
Vänster	`Keyboard.LEFT`
Höger	`Keyboard.RIGHT`
OK eller Välj	`Keyboard.ENTER`

I AIR 2.5 tillkom många andra Keyboard-konstanter för att ge stöd åt indata från fjärrkontroller. Du hittar en fullständig lista i Klassen Keyboard i *Referenshandbok för ActionScript 3.0 i Adobe Flash-plattformen*.

För att försäkra dig om att ditt program fungerar på så många enheter som möjligt bör du tänka på följande:

- Använd bara styrknapparna på fjärrkontrollen, om det är möjligt.

  Olika fjärrkontroller har olika knappar. De flesta har emellertid alltid dessa styrknappar.

  En fjärrkontroll till en Blu-ray-spelare har till exempel ofta inte knappar för en kanal uppåt eller en kanal nedåt. Alla fjärrkontroller har inte ens knappar för uppspelning, paus och stopp.

- Använd knapparna för Meny och Info om programmet har behov av fler knappar än styrknapparna.

  Knapparna för Meny och Info är de näst vanligaste på fjärrkontroller.

- Tänk till exempel på hur vanligt det är med universalfjärrkontroller.

  Även om du skapar ett program för en viss enhet bör du vara medveten om att många inte använder den fjärrkontroll som medföljer enheten. I stället använder de en universalfjärrkontroll. Dessutom programmerar användare inte alltid universalfjärrkontrollen så att knapparna matchar knapparna på enhetens fjärrkontroll. Därför är det bäst att bara använda de vanligaste knapparna.

- Se till att användaren alltid kan komma ur en situation med hjälp av en av styrknapparna.

  Ibland kan det dock finnas en orsak till att ett program använder en av de mer ovanliga knapparna på fjärrkontrollen. Om du skapar en avslutningsmöjlighet med en av styrknapparna kommer programmet att fungera bra på alla enheter.

- Inga pekarindata krävs såvida du inte känner till att målenheten för AIR for TV har pekarfunktionalitet.

  Trots att musindata förväntas i många datorprogram, är det inte vanligt att tv-apparater har stöd för pekarindata. Detta innebär att om du konverterar ett datorprogram för att köras på tv-apparater bör du försäkra dig om att du ändrar programmet så att inga musindata kommer att förväntas. De här ändringarna omfattar ändringar av händelsehantering och av instruktioner till användaren. När programmets startbild visas bör du till exempel inte inkludera text som ”Klicka här för att starta”.

Hantera fokus

När ett element i användargränssnittet har fokus i ett skrivbordsprogram är det målet för användarindatahändelser, som tangentbords- och mushändelser. Ett program markerar dessutom det element i användargränssnittet som har fokus. Hur du hanterar fokus i AIR for TV-program skiljer sig från hur du hanterar fokus i skrivbordsprogram eftersom:

- I skrivbordsprogram används ofta tabbtangenten för att ändra fokus till nästa element i användargränssnittet. Tabbtangenten är inte aktuell i AIR for TV-program. Fjärrkontroller har vanligtvis ingen tabbtangent. Därför är det inte heller aktuellt att hantera fokus med egenskapen `tabEnabled` för ett DisplayObject, som på skrivbordet.
- I skrivbordsprogram förväntas användaren ofta använda musen för att ge fokus åt ett visst element i användargränssnittet.

I ditt program gör du därför följande:

- Lägg till en händelseavlyssnare, som lyssnar efter tangentbordshändelser som `KeyboardEvent.KEY_DOWN`, i Stage-objektet.
- Skapa programlogik för att avgöra vilket element i användargränssnittet som ska markeras för användaren. Se till att ett element i användargränssnittet markeras när programmet startar.
- Baserat på programlogiken skickar du den tangentbordshändelse som scenen tog emot till lämpligt elementobjekt i användargränssnittet.

  Du kan också använda `Stage.focus` eller `Stage.assignFocus()` för att ge fokus åt ett element i användargränssnittet. Du kan sedan lägga till en händelseavlyssnare i aktuellt DisplayObject så att det tar emot tangentbordshändelser.

Utformning av användargränssnittet

Skapa ett användargränssnitt för AIR for TV-program som fungerar bra på tv-apparater genom att följa de här rekommendationerna om:

- programmets svarstider
- programmets användbarhet
- användarens personlighet och förväntningar

Svarstider

Använd de här tipsen för att göra ett AIR for TV-program med så bra svarstider som möjligt.

- Gör programmets SWF-startfil så liten som möjligt.

  Läs bara in de resurser som krävs för att starta programmet i SWF-startfilen. Läs till exempel bara in programmets startbild.

  Den här rekommendationen gäller AIR-skrivbordsprogram, men är ännu viktigare på AIR for TV-enheter. AIR for TV-enheter har till exempel inte samma processorkapacitet som stationära datorer. Dessutom lagras programmet i Flash-minne på dem, vilket inte går lika fort att komma åt som hårddiskar på stationära datorer.

- Se till att programmet körs med en bildrutefrekvens på minst 20 bildrutor per sekund.

  Utforma grafiken för att uppnå detta. Grafikåtgärdernas komplexitet kan påverka antalet bildrutor per sekund. Tips om hur du kan förbättra återgivningsprestanda finns i Optimera prestanda för Adobe Flash-plattformar.

  Obs! *Grafikmaskinvaran på AIR for TV-enheter uppdaterar vanligen skärmen med en frekvens på 60 Hz eller 120 Hz (60 eller 120 gånger per sekund). Maskinvaran skannar scenen efter uppdateringar med exempelvis 30 bildrutor per sekund eller 60 bildrutor per sekund för 60 Hz- eller 120 Hz-skärmar. Om användaren upplever de här högre bildrutefrekvenserna beror däremot på komplexiteten på programmets grafikåtgärder.*

- Uppdatera skärmen inom 100–200 millisekunder efter indata från användaren.

  Användarna blir otåliga om uppdateringen tar längre tid än så, vilket ofta ger upphov till fler knapptryckningar.

Användbarhet

De som använder AIR for TV-program befinner sig i en vardagsrumsmiljö. De sitter på andra sidan rummet från teven, på ungefär tre meters avstånd. Ibland är rummet mörkt. De använder oftast en fjärrkontroll för att styra enheten. Mer än en person kan använda programmet, ibland flera stycken tillsammans och ibland efter varandra.

Därför bör du tänka på följande när du skapar ett användargränssnitt som ska vara användbart på en tv:

- Skapa gränssnittselement som är stora.

  När du skapar text, knappar eller andra element i användargränssnittet bör du komma ihåg att användaren oftast sitter på andra sidan rummet. Se till att allt är lätt att se och läsa på exempelvis tre meters avstånd. Fall inte för frestelsen att fylla skärmen med element bara för att den är stor.

- Använd tydlig konstrast så att innehållet är enkelt att se och läsa från andra sidan rummet.
- Ange tydligt vilket gränssnittselement som har fokus genom att markera det.
- Använd bara rörelse om det behövs. Det kan till exempel fungera bra att glida över från en bild till nästa för kontinuitetens skull. Men rörelser kan vara distraherande om de inte hjälper användaren att navigera eller är nödvändiga för att programmet ska fungera som tänkt.
- Skapa alltid ett tydligt sätt för användaren att gå tillbaka i användargränssnittet.

Mer information om fjärrkontrollen finns i ”Indatahantering via fjärrkontroll” på sidan 128.

Användarens personlighet och förväntningar

Tänk på att de som använder AIR for TV-program oftast är ute efter kvalitetsunderhållning på tv i en rolig och avslappnad miljö. De har inte nödvändigtvis särskilt stor kunskap om datorer eller teknik.

Därför bör du utforma AIR for TV-program med följande i åtanke:

- Använd inte tekniska termer.
- Undvik modala dialogrutor.

- Använd vänliga, informella anvisningar som passar för hemmamiljön, och inte för kontorsmiljöer eller tekniska miljöer.
- Använd grafik med den höga kvalitet som tv-tittare förväntar sig.
- Skapa ett användargränssnitt som fungerar väl tillsammans med en fjärrkontroll. Använd inte användargrässnitt eller designelement som är mer lämpliga för program i en dator eller mobil. Användargränssnitt på datorer och mobila enheter innehåller exempelvis knappar som du ska peka eller klicka på med en mus eller ett finger.

Teckensnitt och text

Du kan använda antingen enhetsteckensnitt eller inbäddade teckensnitt i AIR for TV-program.

Enhetsteckensnitt är teckensnitt som är installerade på en enhet. Alla AIR for TV-enheter har följande enhetsteckensnitt:

Teckensnittsnamn	**Beskrivning**
`_sans`	Enhetsteckensnittet `_sans` är ett teckensnitt av typen sans-serif. Det `_sans`-enhetsteckensnitt som finns på alla AIR for TV-enheter är Myriad Pro. På grund av avståndet är vanligtvis ett sans-serif-teckensnitt bättre på en tv än ett serif-teckensnitt.
`_serif`	Enhetsteckensnittet `_serif` är ett teckensnitt av typen serif. Det `_serif`-enhetsteckensnitt som finns på alla AIR for TV-enheter är Minion Pro.
`_typewriter`	Enhetsteckensnittet `_typewriter` är ett teckensnitt med fast teckenbredd. Det `_typewriter`-enhetsteckensnitt som finns på alla AIR for TV-enheter är Courier Std.

Alla AIR for TV-enheter har dessutom följande asiatiska enhetsteckensnitt:

Teckensnittsnamn	**Språk**	**Teckensnittskategori**	**lokal kod**
RyoGothicPlusN-Regular	Japanska	sans	ja
RyoTextPlusN-Regular	Japanska	serif	ja
AdobeGothicStd-Light	Koreanska	sans	ko
AdobeHeitiStd-Regular	Förenklad kinesiska	sans	zh_CN
AdobeSongStd-Light	Förenklad kinesiska	serif	zh_CN
AdobeMingStd-Light	Traditionell kinesiska	serif	zh_TW och zh_HK

Dessa AIR for TV-enhetsteckensnitt:

- Kommer från Adobe® Type Library
- Ser bra ut på tv-apparater
- Är utformade för videotextning
- Är teckensnittskonturer, inte bitmappteckensnitt

Obs! *Enhetstillverkare inkluderar ofta andra enhetsteckensnitt på enheter. De här tillverkarteckensnitten installeras utöver AIR for TV-enhetsteckensnitten.*

Adobe har ett program som heter FontMaster Deluxe, som visar alla enhetsteckensnitt på enheten. Du kan hämta programmet på Flash Platform for TV.

Du kan också bädda in teckensnitt i AIR for TV-program. Information om inbäddade teckensnitt finns i Avancerad textåtergivning i *Utvecklarhandbok för ActionScript 3.0*.

När det gäller TLF-textfält bör du tänka på följande:

- Använd TLF-textfält för asiatisk text för att utnyttja det språkområde som programmet används i. Ange egenskapen `locale` för det TextLayoutFormat-objekt som är associerat med TLFTextField-objektet. Om du vill veta hur du fastställer det aktuella språkområdet läser du Välja en språkinställning i *Utvecklarhandbok för ActionScript 3.0*.
- Ange teckensnittsnamnet i egenskapen `fontFamily` i TextLayoutFormat-objektet om teckensnittet inte är ett AIR for TV-enhetsteckensnitt. AIR for TV använder teckensnittet om det är tillgängligt på enheten. Om det teckensnitt du begär inte finns på enheten, baserat på inställningen för `locale`, används lämpligt AIR for TV-enhetsteckensnitt i stället automatiskt.
- Ange `_sans`, `_serif` eller `_typewriter` för egenskapen `fontFamily`, tillsammans med inställningen för egenskapen `locale`, för att få AIR for TV att välja korrekt AIR for TV-enhetsteckensnitt. Beroende på språkområdet väljer AIR for TV bland de asiatiska enhetsteckensnitten eller bland de icke-asiatiska enhetsteckensnitten. Med de här inställningarna är det enkelt att automatiskt få rätt teckensnitt för de fyra största asiatiska språkområdena och för engelska.

Obs! *Om du använder klassiska textfält för asiatisk text anger du teckensnittsnamnet för ett AIR for TV-enhetsteckensnitt för att försäkra dig om att det återges korrekt. Om du vet att ett annat teckensnitt är installerat på målenheten kan du även ange det.*

När det gäller programmets prestanda bör du tänka på följande:

- Klassiska textfält har högre prestanda än TLF-textfält.
- Ett klassiskt textfält som använder bitmappteckensnitt har allra bästa prestanda.

  Bitmappteckensnitt har en bitmapp för varje tecken, till skillnad från konturteckensnitt, som bara har konturdata för varje tecken. Både enhetsteckensnitt och inbäddade teckensnitt kan vara bitmappteckensnitt.
- Om du anger ett enhetsteckensnitt försäkrar du dig om att enhetsteckensnittet är installerat på målenheten. Om det inte är installerat på enheten söker AIR for TV efter ett installerat teckensnitt och använder det i stället. Detta beteende påverkar emellertid programmets prestanda negativt.
- Precis som med andra visningsobjekt anger du objektets `cacheAsBitmap`-egenskap som `true`, om TextField-objektet är huvudsakligen oföränderligt. Den här inställningen förbättrar prestanda för övergångar som toningar, glidningar och alfablandningar. Använd `cacheAsBitmapMatrix` för skalning och översättning. Du hittar mer information i ”Maskinvaruacceleration för grafik” på sidan 122.

Filsystemsäkerhet

AIR for TV-program är AIR-program och kan därför komma åt enhetens filsystem. På ”vardagsrumsenheter” är det dock oerhört viktigt att programmet inte kan komma åt enhetens systemfiler eller filer som tillhör andra program. De som använder tv-apparater och associerade enheter förväntar sig inte – och tolererar inte – några enhetsfel. De tittar ju bara på tv.

Ett AIR for TV-program har därför en begränsad vy över enhetens filsystem. Via ActionScript 3.0 kan programmet komma åt enbart specifika kataloger (och deras underkataloger). De katalognamn du använder i ActionScript är dessutom inte de verkliga katalognamnen som används på enheten. Det här extra lagret ser till AIR for TV-program inte kommer åt lokala filer som de inte äger, vare sig det är av misstag eller av andra skäl.

Du hittar mer information i Katalogvy för AIR for TV-program.

AIR-programsandlådan

AIR for TV-program körs i AIR-programsandlådan, som beskrivs i AIR-programsandlådan.

Den enda skillnaden för AIR for TV-program är att de har begränsad tillgång till filsystemet, enligt beskrivningen i ”Filsystemsäkerhet” på sidan 132.

Programmets livscykel

Till skillnad från hur det fungerar på en stationär dator kan slutanvändaren inte stänga det fönster i vilket ditt AIR for TV-program körs. Därför bör du alltid skapa en funktion för att avsluta programmet i användargränssnittet.

Vanligtvis kan slutanvändaren alltid avsluta ett program på enheten med Exit-knappen på fjärrkontrollen. AIR for TV skickar emellertid inte händelsen `flash.events.Event.EXITING` till programmet. Därför bör du spara programmets status ofta, så att det kan återställas till ett rimligt läge nästa gång det startas.

HTTP-cookies

AIR for TV har stöd för HTTP-beständiga cookies och sessions-cookies. I AIR for TV lagras varje cookies för AIR-program i en programspecifik katalog:

`/app-storage/<app id>/Local Store`

Cookie-filnamnet är `cookies`.

Obs! *AIR på andra enheter, till exempel på en datorenhet, lagrar inte cookies separat för varje program. Programspecifika cookie-lager ger stöd åt programmet och systemsäkerhetsmodellen för AIR for TV.*

Använd ActionScript-egenskapen `URLRequest.manageCookies` enligt följande:

- Ange `manageCookies` som `true`. Detta är standardvärdet. Det betyder att AIR for TV automatiskt lägger till cookies i HTTP-begäranden och kommer ihåg cookies i HTTP-svar.

  Obs! *Även om `manageCookies` är `true` kan programmet lägga till en cookie i en HTTP-begäran manuellt med hjälp av `URLRequest.requestHeaders`. Om den här cookie-filen har samma namn som en cookie-fil som hanteras av AIR for TV innehåller begäran två cookie-filer med samma namn. Värdena för de båda cookie-filerna kan vara olika.*

- Ange `manageCookies` som `false`. Det här värdet innebär att programmet ansvarar för att skicka cookies i HTTP-begäranden och för att komma ihåg cookies i HTTP-svar.

Du hittar mer information i URLRequest.

Arbetsflöde för att utveckla ett AIR for TV-program

Du kan utveckla AIR for TV-program med följande utvecklingsverktyg för Adobe Flash-plattformen:

- Adobe Flash Professional

  Adobe Flash Professional CS5.5 har stöd för AIR 2.5 for TV, den första AIR-versionen med stöd för AIR for TV-program.

- Adobe Flash® Builder®

  Flash Builder 4.5 har stöd för AIR 2.5 for TV.

- AIR SDK

Från och med AIR 2.5 kan du utveckla program med de kommandoradsverktyg som ingår i AIR SDK. Om du vill hämta AIR SDK går du till http://www.adobe.com/se/products/air/sdk/.

Använda Flash Professional

Du utvecklar, testar och publicerar AIR for TV-program med Flash Professional på ungefär samma sätt som vanliga AIR-datorprogram.

När du skriver ActionScript 3.0-kod bör du emellertid bara använda de klasser och metoder som AIR-profilerna `tv` och `extendedTV` har stöd för. Mer information finns i ”Enhetsprofiler” på sidan 241.

Projektinställningar

Gör följande för att konfigurera ditt projekt för ett AIR for TV-program:

- Ange värdet för Spelare som minst är AIR 2.5 på fliken Flash i dialogrutan Publiceringsinställningar.
- Välj profilen `TV` eller `extendedTV` på fliken Allmänt i dialogrutan Inställningar i Adobe AIR.

Felsökning

Du kan köra programmet med AIR Debug Launcher inifrån Flash Professional. Gör så här:

- Om du vill köra programmet i felsökningsläge väljer du:

  Felsök > Felsök filmen > I AIR Debug Launcher (skrivbord)

  När du har valt detta kan du för efterföljande felsökningar välja:

  Felsök > Felsök filmen > Felsök

- Om du vill köra programmet utan funktioner för felsökning väljer du:

  Kontroll > Testa filmen > I AIR Debug Launcher (skrivbord)

  När du har valt detta kan du för efterföljande körningar välja Kontroll > Testa filmen > Testa.

Eftersom du har angett AIR-profilen som tv eller extendedTV visas en meny som heter Remote Control Buttons i AIR Debug Launcher. Du kan använda den här menyn för att simulera knapptryckningar på en fjärrkontroll.

Du hittar mer information i ”Fjärrfelsökning med Flash Professional” på sidan 142.

Använda ANE-tillägg

Om ditt program använder ett ANE-tillägg följer du anvisningarna i ”Uppgiftslista för att använda ANE-tillägg” på sidan 147.

När ett program använder ANE-tillägg bör tänka på följande:

- Du kan inte publicera programmet med Flash Professional. Om du vill publicera programmet använder du ADT. Läs mer i ”Paketera med ADT” på sidan 139.
- Du kan inte köra eller felsöka programmet med Flash Professional. Om du vill felsöka programmet på utvecklingsdatorn använder du ADL. Läs mer i ”Enhetssimulering med ADL” på sidan 140.

Använda Flash Builder

Från och med Flash Builder 4.5 har Flash Builder stöd för AIR for TV-utveckling. Du utvecklar, testar och publicerar AIR for TV-program med Flash Builder på ungefär samma sätt som vanliga AIR-skrivbordsprogram.

Konfigurera programmet

Se till att programmet gör följande:

- Använder `Application`-elementet som behållarklass i MXML-filen, om du använder en MXML-fil:

```
<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>.
```

  Viktigt! *AIR for TV-program har inte stöd för elementet* `WindowedApplication`*.*

  Obs! *Du behöver inte använda en MXML-fil alls. I stället kan du skapa ett ActionScript 3.0-projekt.*

- Bara använder ActionScript 3.0-klasser och -metoder som AIR-profilerna `tv` och `extendedTV` har stöd för. Mer information finns i ”Enhetsprofiler” på sidan 241.

Dessutom ser du till att följande krav uppfylls i programmets XML-fil:

- `application`-elementets `xmlns`-attribut är inställt på AIR 2.5:

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

- Elementet `supportedProfiles` inkluderar `tv` eller `extendedTV`:

```
<supportedProfiles>tv</supportedProfiles>
```

Felsöka programmet

Du kan köra programmet med AIR Debug Launcher inifrån Flash Builder. Gör så här:

1. Välj Run > Debug Configurations.
2. Kontrollera att fältet Profile är inställt på Desktop.
3. Välj Run > Debug om du vill köra i felsökningsläge eller välj Run > Run om du vill köra utan funktioner för felsökning.

Eftersom du har angett elementet `supportedProfiles` som tv eller extendedTV visas en meny som heter Remote Control Buttons i AIR Debug Launcher. Du kan använda den här menyn för att simulera knapptryckningar på en fjärrkontroll.

Du hittar mer information i ”Fjärrfelsökning med Flash Builder” på sidan 143.

Använda ANE-tillägg

Om ditt program använder ett ANE-tillägg följer du anvisningarna i ”Uppgiftslista för att använda ANE-tillägg” på sidan 147.

När ett program använder ANE-tillägg bör tänka på följande:

- Du kan inte publicera programmet med Flash Builder. Om du vill publicera programmet använder du ADT. Läs mer i ”Paketera med ADT” på sidan 139.
- Du kan inte köra eller felsöka programmet med Flash Builder. Om du vill felsöka programmet på utvecklingsdatorn använder du ADL. Läs mer i ”Enhetssimulering med ADL” på sidan 140.

Egenskaper i programbeskrivningsfilen för AIR for TV

Precis som i andra AIR-program anger du de grundläggande programegenskaperna i programbeskrivningsfilen. I program i tv-profilen ignoreras vissa skrivbordsspecifika egenskaper, som fönsterstorlek och genomskinlighet. Program som har enheter i profilen `extendedTV` som mål kan använda ANE-tillägg. Dessa program identifierar de ANE-tillägg som används i ett `extensions`-element.

Vanliga inställningar

Det finns flera programbeskrivningsinställningar som är viktiga för alla program i tv-profilen.

Nödvändig version av AIR-miljön

Ange den version av AIR-miljön som krävs för ditt program med hjälp av namnutrymmet i programbeskrivningsfilen.

Namnutrymmet, som tilldelas i elementet `application`, avgör till stor del vilka funktioner ditt program kan använda. Tänk dig till exempel ett program som använder AIR 2.5-namnutrymmet, men där användaren har en senare version installerad. I så fall ser ditt program AIR 2.5-beteendet, även om beteendet har ändrats i den senare AIR-versionen. Ditt program får inte tillgång till de nya beteendena och funktionerna förrän du ändrar namnutrymmet och publicerar en uppdatering. Säkerhetskorrigeringar är viktiga undantag till den här regeln.

Ange namnutrymmet med `xmlns`-attributet för rotelementet `application`:

```
<application xmlns="http://ns.adobe.com/air/application/2.5">
```

AIR 2.5 är den första versionen av AIR som har stöd för tv-program.

Programidentitet

Det är flera inställningar som bör vara unika för varje program du publicerar. Bland dessa inställningar finns elementen `id`, `name` och `filename`.

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

Programversion

Ange programversionen i elementet `versionNumber`. När du anger ett värde för `versionNumber` kan du använda en sekvens med upp till tre siffror, åtskilda med punkter, t.ex. ”0.1.2”. Varje segment i versionsnumret kan ha upp till tre siffror. (Följaktligen är ”999.999.999” det största versionsnummer som tillåts.) Du behöver inte inkludera alla tre segment i versionsnumret; "1" och "1.0" är också giltiga versionsnummer.

Du kan också ange en etikett för versionen med elementet `versionLabel`. När du lägger till en versionsetikett visas den i stället för versionsnumret.

```
<versionNumber>1.23.7<versionNumber>
<versionLabel>1.23 Beta 7</versionLabel>
```

Huvudsaklig SWF-programfil

Ange den huvudsakliga SWF-programfilen i det underordnade `versionLabel`-elementet för `initialWindow`-elementet. När du har enheter i tv-profilen som mål måste du använda en SWF-fil (HTML-baserade program stöds inte).

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

Du måste inkludera filen i AIR-paketet (med ADT eller den integrerade utvecklingsmiljön). Det räcker inte att bara referera till namnet i programbeskrivningen för att filen automatiskt ska inkluderas i paketet.

Egenskaper för huvudskärmen

Flera underordnade element till elementet `initialWindow` styr det inledande utseendet och beteendet för huvudprogramskärmen. De flesta av dessa egenskaper ignoreras på enheter i tv-profilerna, men du kan använda elementet `fullScreen`:

- **`fullScreen`** – Anger om programmet får uppta hela enhetens skärm eller om det ska dela skärmen med vanliga systemkontroller.

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

Elementet visible

Elementet `visible` är ett underordnat element till `initialWindow`-elementet. I AIR for TV ignoreras `visible`-elementet eftersom programmets innehåll alltid visas på AIR for TV-enheter.

Du kan emellertid ställa in `true` för `visible`-elementet om programmet även ska användas på datorer.

På datorer är elementets standardvärde `false`. Om du därför inkluderar elementet `visible` kommer programinnehållet att visas på en dator. Trots att du kan använda ActionScript-klassen NativeWindow för att göra innehållet synligt på datorer, har tv-enhetsprofilerna inte stöd för klassen NativeWindow. Programmet kommer inte att läsas in om du försöker använda klassen NativeWindow i ett program som körs på en AIR for TV-enhet. Det har ingen betydelse om du anropar en metod i klassen NativeWindow; ett program som använder klassen kommer inte att läsas in på AIR for TV-enheter.

Profiler som stöds

Om ditt program bara är avsett för tv-enheter kan du förhindra att det kan installeras på andra sorters enheter. Uteslut andra profiler från listan över profiler som stöds i elementet `supportedProfiles`:

```
<supportedProfiles>tv extendedTV</supportedProfiles>
```

Om ett program använder ett ANE-tillägg inkluderar du endast profilen `extendedTV` i listan över profiler som stöds:

```
<supportedProfiles>extendedTV</supportedProfiles>
```

Om du utelämnar elementet `supportedProfiles` antas programmet ha stöd för alla profiler.

Inkludera inte *enbart* `tv`-profilen i `supportedProfiles`-listan. Vissa tv-enheter kör alltid AIR for TV i ett läge som motsvarar profilen `extendedTV`. Detta beteende innebär att AIR for TV kan köra program där ANE-tillägg används. Om `supportedProfiles`-elementet endast anger `tv`, innebär det att innehållet inte är kompatibelt med AIR for TV-läget för `extendedTV`. Därför kommer vissa tv-apparater inte att läsa in program som endast anger `tv`-profilen.

En lista över de ActionScript-klasser som stöds i profilerna `tv` och `extendedTV` finns i ”Funktioner i de olika profilerna” på sidan 242.

Obligatoriska ANE-tillägg

Program som har stöd för profilen `extendedTV` kan använda ANE-tillägg.

Deklarera alla ANE-tillägg som används av AIR-programmet i programbeskrivningen med hjälp av elementen `extensions` och `extensionID`. Följande exempel visar syntaxen för att ange två nödvändiga ANE-tillägg:

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

Programmet kan endast visa tillägg som visas i listan.

Elementet `extensionID` har samma värde som elementet `id` i tilläggsbeskrivningsfilen. Tilläggsbeskrivningsfilen är en XML-fil med namnet extension.xml. Den är paketerad i den ANE-fil som du får från enhetstillverkaren.

Programmet kommer inte att köras om du listar ett tillägg i `extensions`-elementet och AIR for TV-enheten inte har tillägget installerat. Undantaget till denna regel är om ANE-filen som paketeras med AIR for TV-programmet har en stub-version av tillägget. Om det är så kan programmet köras och stub-version av tillägget kommer då att användas. Stub-versionen innehåller ActionScript-kod, men ingen ANE-kod.

Programikoner

Kraven för programikoner på tv-enheter är enhetsspecifika. Enhetens tillverkare kan till exempel ange:

- Nödvändiga ikoner och ikonstorlekar.
- Nödvändiga filtyper och namnkonventioner.
- Hur ikonerna för programmet ska tillhandahållas, till exempel om de ska paketeras med programmet.
- Om ikonerna ska anges i ett `icon`-element i programbeskrivningsfilen.
- Beteende om programmet inte har några ikoner.

Kontakta enhetstillverkaren för att få mer information.

Ignorerade inställningar

Program på tv-enheter ignorerar programinställningar som gäller mobilfunktioner, funktioner för systemfönster eller funktioner i operativsystem på stationära datorer. De inställningar som ignoreras är:

- allowBrowserInvocation
- aspectRatio
- autoOrients
- customUpdateUI
- fileTypes
- height
- installFolder
- maximizable
- maxSize
- minimizable
- minSize
- programMenuFolder
- renderMode
- resizable
- systemChrome

- title
- transparent
- visible
- width
- x
- y

Paketera ett AIR for TV-program

Paketera med ADT

Du kan använda kommandoradsverktyget ADT i AIR för att paketera ett AIR for TV-program. Från och med AIR SDK version 2.5, har ADT stöd för paketering för tv-enheter. Innan du paketerar programmet kompilerar du all ActionScript-kod och MXML-kod. Du måste också ha ett kodsigneringscertifikat. Du kan skapa ett certifikat med ADT-kommandot -certificate.

Du hittar detaljerad referensinformation om ADT-kommandon och -alternativ i ”AIR Developer Tool (ADT)” på sidan 162.

Skapa ett AIR-paket

Använd ADT-kommandot package för att skapa ett AIR-paket:

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

Exemplet förutsätter följande:

- Att sökvägen till ADT-verktyget finns i kommandoradens sökvägsdefinition. (Läs mer i ”Systemvariabeln path” på sidan 299.)
- Certifikatet codesign.p12 finns i den katalog som är överordnad den katalog varifrån du kör ADT-kommandot.

Kör kommandot från den katalog som innehåller programfilerna. Programfilerna i exemplet är myApp-app.xml (programbeskrivningsfilen), myApp.swf och en ikonkatalog.

När du kör kommandot så som visas tillfrågas du om lösenordet för nyckelbehållaren i ADT. De lösenordstecken du skriver visas inte i alla skalprogram. Tryck bara på Retur när du är klar. Du kan också använda parametern `storepass` för att inkludera lösenordet i ADT-kommandot.

Skapa ett AIRN-paket

Om ditt AIR for TV-program använder ett ANE-tillägg skapar du ett AIRN-paket i stället för ett AIR-paket. Om du vill skapa ett AIRN-paket använder du ADT-kommandot package och anger måltypen som `airn`.

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp-app.xml
myApp.swf icons -extdir C:\extensions
```

Exemplet förutsätter följande:

- Att sökvägen till ADT-verktyget finns i kommandoradens sökvägsdefinition. (Läs mer i ”Systemvariabeln path” på sidan 299.)
- Certifikatet codesign.p12 finns i den katalog som är överordnad den katalog varifrån du kör ADT-kommandot.
- Parametern `-extdir` namnger en katalog som innehåller de ANE-filer som programmet använder.

Dessa ANE-filer innehåller en ActionScript-version som är en stub eller simulator av tillägget. Den version av tillägget som innehåller den systemspecifika koden installeras på AIR for TV-enheten.

Kör kommandot från den katalog som innehåller programfilerna. Programfilerna i exemplet är myApp-app.xml (programbeskrivningsfilen), myApp.swf och en ikonkatalog.

När du kör kommandot så som visas tillfrågas du om lösenordet för nyckelbehållaren i ADT. De lösenordstecken du skriver visas inte i alla skalprogram. Tryck bara på Retur när du är klar. Du kan också använda parametern `storepass` för att inkludera lösenordet i kommandot.

Du kan också skapa en AIRI-fil för ett AIR for TV-program som använder ANE-tillägg. AIRI-filen är likadan som AIRN-filen, med undantag för att den inte är signerad. Till exempel:

```
adt -prepare myApp.airi myApp.xml myApp.swf icons -extdir C:\extensions
```

Du kan sedan skapa en AIRN-fil från AIRI-filen när du är redo att signera programmet:

```
adt -package -storetype pkcs12 -keystore ../codesign.p12 -target airn myApp.airn myApp.airi
```

Du hittar mer information i Developing Native Extensions for Adobe AIR.

Paketera med Flash Builder eller Flash Professional

Med Flash Professional och Flash Builder kan du publicera eller exportera AIR-paket utan att köra ADT själv. Hur du skapar ett AIR-paket för ett AIR-program beskrivs i dokumentationen för dessa program.

För närvarande är det emellertid endast ADT som kan skapa AIRN-paket, programpaket för AIR for TV-program där ANE-tillägg används.

Mer information finns i följande avsnitt:

- Paketera AIR-program med Flash Builder
- Publicera för Adobe AIR

Felsöka AIR for TV-program

Enhetssimulering med ADL

Det snabbaste och enklaste sättet att testa och felsöka de flesta programfunktioner är att köra programmet på utvecklingsdatorn med ADL-verktyget (Adobe Debug Launcher).

ADL använder elementet `supportedProfiles` i programbeskrivningen för att fastställa vilken profil som ska användas. Närmare bestämt:

- Om det finns fler än en profil används den första i listan.
- Du kan använda parametern `-profile` i ADL för att välja en av de andra profilerna i listan `supportedProfiles`.
- Om du inte inkluderar ett `supportedProfiles`-element i programbeskrivningen kan vilken profil som helst anges för argumentet `-profile`.

Använd till exempel följande kommando för att starta ett program för att simulera `tv`-profilen:

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

När du simulerar `tv`- eller `extendedTV`-profilen på skrivbordet med ADL körs programmet i en miljö som bättre motsvarar en målenhet. Till exempel:

- ActionScript-API:er som inte ingår i profilen i `-profile`-argumentet är inte tillgängliga.
- ADL tillåter indata från enhetskontroller, som fjärrkontroller, via menykommandon.
- Om du anger `tv` eller `extendedTV` i argumentet `-profile` kan ADL simulera klassen StageVideo på skrivbordet.
- Om du anger `extendedTV` i argumentet `-profile` kan programmet använda stubbar eller simulatorer för ANE-tillägg, som paketerats med programmets AIRN-fil.

Men eftersom ADL kör programmet på skrivbordet har det sina begränsningar att testa AIR for TV-program med ADL:

- Det speglar inte programmets prestanda på enheten. Kör prestandatester på målenheten.
- Det simulerar inte begränsningarna för StageVideo-objektet. Vanligtvis använder du klassen StageVideo, inte klassen Video, för att spela upp en video med AIR for TV-enheter som mål. Klassen StageVideo utnyttjar prestandafördelarna med enhetens maskinvara, men har vissa begränsningar när det gäller visning. ADL spelar upp videon på skrivbordet utan de här begränsningarna. Testa därför att spela upp videon på målenheten.
- Det simulerar inte den systemspecifika koden för ett ANE-tillägg. Du kan däremot ange profilen `extendedTV`, som har stöd för ANE-tillägg, i ADL-argumentet `-profile`. Med ADL kan du testa med den ActionScript-version, som är en stub eller simulator av tillägget, som ingår i ANE-paketet. Vanligtvis inkluderar motsvarande tillägg, som är installerat på enheten, även systemspecifik kod. Om du vill testa med tillägget och dess systemspecifika kod, ska du köra programmet på målenheten.

Du hittar mer information i ”AIR Debug Launcher (ADL)” på sidan 156.

Använda ANE-tillägg

Om ditt program använder ANE-tillägg ser ADL-kommandot ut som i följande exempel:

```
adl -profile extendedTV -extdir C:\extensionDirs myApp-app.xml
```

Exemplet förutsätter följande:

- Att sökvägen till ADL-verktyget finns i kommandoradens sökvägsdefinition. (Läs mer i ”Systemvariabeln path” på sidan 299.)
- Att den aktuella katalogen innehåller programfilerna. Bland de här filerna finns SWF-filerna och programbeskrivningsfilen, som i det här exemplet heter myApp-app.xml.
- Parametern `-extdir` namnger en katalog som innehåller en katalog för varje ANE-tillägg som används i programmet. Var och en av katalogerna innehåller den *opaketerade* ANE-filen för ett tillägg. Till exempel:

```
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
```

Dessa opackade ANE-filer innehåller en ActionScript-version som är en stub eller simulator av tillägget. Den version av tillägget som innehåller den systemspecifika koden installeras på AIR for TV-enheten.

Du hittar mer information i Developing Native Extensions for Adobe AIR.

Indata från kontroller

ADL simulerar knapparna på fjärrkontrollen till en tv-enhet. Du kan skicka dessa knappindata till den simulerade enheten med den meny som visas när ADL startas med en av tv-profilerna.

Skärmstorlek

Du kan testa programmet med olika skärmstorlekar genom att ange ADL-parametern `-screensize`. Du kan ange en sträng med fyra värden, som representerar bredden och höjden på den normala och den maximerade skärmen.

Ange alltid pixeldimensioner för stående layout, d.v.s. ange bredden som ett värde som är mindre än höjden. Till exempel:

```
adl -screensize 728x1024:768x1024 myApp-app.xml
```

Trace-programsatser

När du kör tv-programmet på skrivbordet skrivs utdata från trace-programsatser antingen till felsökaren eller till det terminalfönster som användes för att starta ADL.

Fjärrfelsökning med Flash Professional

Du kan använda Flash Professional för att fjärrfelsöka AIR for TV-programmet när det körs på målenheten. Vilka steg som behöver konfigureras för fjärrfelsökningen beror däremot på enheten. Till exempel innehåller Adobe® AIR® for TV MAX 2010 Hardware Development Kit dokumentation med detaljerad information för den enheten.

Utför följande steg för att förbereda för fjärrfelsökning, oavsett målenhet:

1 Markera Tillåt felsökning på fliken Flash i dialogrutan Publiceringsinställningar.

Med det här alternativet inkluderar Flash Professional felsökningsinformation i alla SWF-filer som skapas från FLA-filen.

2 På fliken Signatur i dialogrutan Inställningar i Adobe AIR (AIR – Program- och installationsinställningar) väljer du alternativet att förbereda en AIRI-fil (AIR Intermediate).

Medan du fortfarande utvecklar programmet är det tillräcklig om du använder en AIRI-fil, eftersom ingen digital signatur krävs för den.

3 Publicera programmet, så skapas AIRI-filen.

De sista stegen är att installera och köra programmet på målenheten. De här stegen skiljer sig dock åt på olika enheter.

Fjärrfelsökning med Flash Builder

Du kan också använda Flash Builder för att fjärrfelsöka AIR for TV-programmet när det körs på målenheten. Vilka steg som behövs för fjärrfelsökningen beror däremot på enheten.

Utför följande steg för att förbereda för fjärrfelsökning, oavsett målenhet:

1 Välj Project (Projekt) > Export Release Build (Exportera färdig version). Välj alternativet att förbereda en AIRI-fil (AIR Intermediate).

Medan du fortfarande utvecklar programmet är det tillräcklig om du använder en AIRI-fil, eftersom ingen digital signatur krävs för den.

2 Publicera programmet, så skapas AIRI-filen.

3 Ändra programmets AIRI-paket så att det innehåller SWF-filerna med felsökningsinformation.

De SWF-filer som innehåller felsökningsinformation finns i Flash Builder-projektkatalogen för programmet, i en katalog med namnet bin-debug. Ersätt SWF-filerna i AIRI-paketet med SWF-filerna i katalogen bin-debug.

På en utvecklingsdator med Windows kan du utföra det här bytet på följande sätt:

1 Byt namn på AIRI-paketfilen så att den får filtillägget .zip i stället för .airi.

2 Extrahera innehållet i ZIP-filen.

3 Ersätt SWF-filerna i den extraherade katalogstrukturen med dem som finns i bin-debug.

4 Komprimera filerna i den extraherade katalogen.

5 Ändra tillbaka ZIP-filens namn till filtillägget .airi.

Om du använder en utvecklingsdator med Mac OS utförs det här bytet på olika sätt beroende på enheten. Vanligtvis inkluderar de följande:

1 Installera AIRI-paketet på målenheten.

2 Ersätt SWF-filerna i programmets installationskatalog på målenheten med SWF-filerna från katalogen bin-debug.

Som exempel kan nämnas den enhet som ingår i Adobe AIR for TV MAX 2010 Hardware Development Kit. Installera AIRI-paketet enligt beskrivningen i den dokumentationen. Använd sedan telnet på kommandoraden på Mac-utvecklingsdatorn för att komma åt målenheten. Ersätt SWF-filerna i programmets installationskatalog på /opt/adobe/stagecraft/apps/<*programnamn*>/ med SWF-filerna från katalogen bin-debug.

Följande steg gäller fjärrfelsökning med Flash Builder och den enhet som ingår i Adobe AIR for TV MAX 2010 Hardware Development Kit.

1. På den dator där Flash Builder körs (utvecklingsdatorn) kör du den AIR for TV Device Connector som ingår i MAX 2010 Hardware Development Kit. Den visar IP-adressen för utvecklingsdatorn.
2. På enheten med MAX 2010 Hardware Development Kit startar du programmet DevMaster, som också ingår i det paketet.
3. I programmet DevMaster anger du IP-adressen för utvecklingsdatorn så som den visas i AIR for TV Device Connector.
4. Kontrollera att Enable Remote Debugging är markerat i DevMaster.
5. Avsluta programmet DevMaster.
6. På utvecklingsdatorn väljer du Start i AIR for TV Connector.
7. Starta ett annat program på enheten med MAX 2010 Hardware Development Kit. Kontrollera att spårningsinformation visas i AIR for TV Device Connector.

   Om spårningsinformationen inte visas är utvecklingsdatorn och enheten med MAX 2010 Hardware Development Kit inte anslutna till varandra. Kontrollera att den port på utvecklingsdatorn som används för spårningsinformation är tillgänglig. Du kan välja en annan port i AIR for TV Device Connector. Kontrollera även att brandväggen tillåter åtkomst till den valda porten.

Starta sedan felsökaren i Flash Builder. Gör så här:

1. Gå till Flash Builder och välj Run > Debug Configurations.
2. Kopiera projektets namn från den befintliga felsökningskonfigurationen, som är avsedd för lokal felsökning.
3. Välj Web Application i dialogrutan Debug Configurations. Välj sedan ikonen New Launch Configuration.
4. Klistra in projektnamnet i fältet Project.
5. Avmarkera kryssrutan Use Default i delen URL Or Path To Launch. Skriv även `about:blank` i textfältet.
6. Klicka på Apply för att spara ändringarna.
7. Välj Debug för att starta Flash Builder-felsökaren.
8. Starta programmet på enheten med MAX 2010 Hardware Development Kit.

Nu kan du använda felsökaren i Flash Builder för att till exempel ange brytpunkter och kontrollera variabler.

Kapitel 9: Använda ANE-tillägg för Adobe AIR

ANE-tillägg (AIR Native Extensions) för Adobe AIR ger dig ActionScript-API:er med tillgång till enhetsspecifik funktionalitet programmerade i systemspecifik kod. ANE-utvecklare arbetar ibland med enhetstillverkare och tredjepartsleverantörer.

Om du är en ANE-utvecklare ska du besöka Developing Native Extensions for Adobe AIR.

Ett ANE-tillägg består av en kombination av:

- ActionScript-klasser.
- Systemspecifik kod.

Om du utvecklar AIR-program och använder ANE-tillägg, kommer du endast att arbeta med ActionScript-klasserna.

ANE-tillägg är användbart i följande situationer.

- Den systemspecifika kodimplementationen erbjuder åtkomst till plattformsspecifika funktioner. Dessa plattformsspecifika funktioner är inte tillgängliga i de interna ActionScript-klasserna och de kan inte implementeras i programspecifika ActionScript-klasser. Den systemspecifika kodimplementationen kan erbjuda sådan funktionalitet eftersom den har åtkomst till enhetsspecifik maskin- och programvara.
- En systemspecifik implementation kan i vissa tillfällen vara snabbare än implementeringar där enbart ActionScript används.
- Den systemspecifika kodimplementationen kan ge dig ActionScript-åtkomst till äldre systemspecifik kod.

Några exempel på ANE-tillägg hittar du i Adobe Developer Center. Ett ANE-tillägg kan till exempel ge AIR-programåtkomst till vibratorfunktionen för Android. Se ANE-tillägg för Adobe AIR.

ANE-filer (AIR Native Extension)

ANE-utvecklare paketerar ANE-tilläggen i en ANE-fil. En ANE-fil är en arkivfil som innehåller de nödvändiga biblioteken och resurserna för ANE-tillägget.

Tänk på att för vissa enheter innehåller ANE-filen det systemspecifika kodbiblioteket som används i ANE-tilläggen. För andra enheter installeras emellertid det systemspecifika kodbiblioteket på enheten. I vissa fall saknar ANE-tillägget systemspecifik kod för en speciell enhet, eftersom den endast implementeras med ActionScript.

Om du är AIR-utvecklare använder du ANE-filer på följande sätt:

- Lägg till ANE-filen i programmets bibliotekssökväg på samma sätt som du inkluderar en SWC-fil i bibliotekssökvägen. Detta gör att programmet kan referera till tilläggets ActionScript-klasser.

  Obs! *När du kompilerar programmen ska du se till att använda dynamiska länkar för ANE-filen. Om du använder Flash Builder ska du ange External på egenskapspanelen i ActionScript Builder Path; om du använder kommandoraden anger du -external-library-path.*

- Paketera ANE-filen med AIR-programmet.

ANE-tillägg och ActionScript-klassen NativeProcess

I ActionScript 3.0 finns en NativeProcess-klass. Klassen gör att ett AIR-program kan köra systemspecifika processer i värdoperativsystemet. Detta påminner om ANE-tillägg som ger åtkomst till plattformsspecifika funktioner och bibliotek. När du väljer mellan att använda klassen NativeProcess eller ett ANE-tillägg ska du tänka på följande:

- Endast AIR-profilen `extendedDesktop` har stöd för klassen NativeProcess. ANE-tillägg är därför det enda alternativet för program med AIR-profilerna , `mobileDevice` och `extendedMobileDevice`.
- ANE-utvecklare skapar ofta systemspecifika implementationer för olika plattformar, men det ActionScript-API som används är vanligtvis detsamma för alla plattformar. När klassen NativeProcess används, kan den ActionScript-kod som ska inleda den systemspecifika processen variera mellan olika plattformar.
- Klassen NativeProcess startar en separat process, medan ett ANE-tillägg körs i samma process som AIR-programmet. Detta innebär, om du är orolig för att koden ska krascha, att du ska använda klassen NativeProcess eftersom den är säkrare. Olika processer innebär emellertid att du troligtvis måste implementera kommunikationshantering mellan processerna.

ANE-tillägg och ActionScript-klassbibliotek (SWC-filer)

En SWC-fil är ett ActionScript-klassbibliotek i ett arkivformat. SWC-filen innehåller en SWF-fil och ytterligare resursfiler. SWC-filer är ett bekvämt sätt att dela ActionScript-klasser i stället för att dela enskild ActionScript-kod och resursfiler.

Ett systemspecifikt tilläggspaket är en ANE-fil. Precis som en SWC-fil, är en ANE-fil även ett ActionScript-klassbibliotek med en SWF-fil och andra resursfiler i ett arkivformat. Den viktigaste skillnaden mellan en ANE-fil och en SWC-fil är att en ANE-fil kan innehålla ett systemspecifikt kodbibliotek.

Obs! *När du kompilerar programmen ska du se till att använda dynamiska länkar för ANE-filen. Om du använder Flash Builder ska du ange External på egenskapspanelen i ActionScript Builder Path; om du använder kommandoraden anger du -external-library-path.*

Fler hjälpavsnitt

Om SWC-filer

Enheter som stöds

Från och med AIR 3 kan du använda ANE-tillägg i program för följande enheter:

- Android-enheter, från och med Android 2.2
- iOS-enheter, från och med iOS 4.0
- iOS-simulator, från och med AIR 3.3
- Blackberry Playbook
- Windows-datorer med stöd för AIR 3.0
- Mac OS X-datorer med stöd för AIR 3.0

Det går ofta att använda samma ANE-tillägg för flera plattformar. Tilläggets ANE-fil innehåller ActionScript och systemspecifika bibliotek för varje plattform som stöds. Vanligtvis används samma publika gränssnitt för ActionScript-bibliotek på alla plattformar. De systemspecifika biblioteken är annorlunda.

Ibland har ANE-tillägg stöd för en standardplattform. Implementationen av standardplattformen innehåller endast ActionScript-kod, ingen systemspecifik kod. Om du paketerar ett program för en plattform som tillägget inte direkt har stöd för, kommer programmet att använda standardimplementationen när det körs. Tänk dig till exempel ett tillägg med en funktion som endast gäller för mobila enheter. Tillägget kan även tillhandahålla en standardimplementation som ett datorprogram kan använda för att simulera funktionen.

Enhetsprofiler som stöds

Följande AIR-profiler har stöd för ANE-tillägg:

- `extendedDesktop`, från och med AIR 3.0
- `mobileDevice`, från och med AIR 3.0
- `extendedMobileDevice`, från och med AIR 3.0

Fler hjälpavsnitt

Stöd för AIR-profil

Uppgiftslista för att använda ANE-tillägg

Gör följande för att använda ANE-tillägg i dina program:

1. Deklarera tillägget i programbeskrivningsfilen.
2. Inkludera ANE-filen i programmets bibliotekssökväg.
3. Paketera programmet.

Deklarera tillägg i programbeskrivningsfiler

Alla AIR-program har en programbeskrivningsfil. När ett program använder ANE-tillägg, kommer programbeskrivningsfilen att inkludera ett `<extensions>`-element. Till exempel:

```
<extensions>
    <extensionID>com.example.Extension1</extensionID>
    <extensionID>com.example.Extension2</extensionID>
</extensions>
```

Elementet `extensionID` har samma värde som elementet `id` i tilläggsbeskrivningsfilen. Tilläggsbeskrivningsfilen är en XML-fil med namnet extension.xml. Den paketeras i en ANE-fil. Du kan använda ett arkivextraheringsverktyg för att läsa extension.xml-filen.

Inkludera ANE-filen i programmets bibliotekssökväg

Inkludera ANE-filen i bibliotekssökvägen när du kompilerar ett program som använder ett ANE-tillägg.

Använda ANE-filen med Flash Builder

Om ditt program använder ett ANE-tillägg inkluderar du ANE-filen för tillägget i bibliotekssökvägen. Sedan kan du använda Flash Builder för att kompilera ActionScript-koden.

Utför följande steg med hjälp av Flash Builder 4.5.1:

1. Ändra filnamnstillägget på ANE-filen från .ane till .swc. Detta steg är nödvändigt för att Flash Builder ska kunna hitta filen.
2. Välj Project > Properties i Flash Builder-projektet.
3. Välj Flex Build Path i dialogrutan Properties.
4. Välj Add SWC... på fliken Library Path.
5. Bläddra till SWC-filen och klicka på Öppna.
6. Klicka på OK i dialogrutan Add SWC....

   ANE-filen visas nu på fliken Library Path i dialogrutan Properties.
7. Utöka posten för SWC-filen. Dubbelklicka på Link Type för att öppna dialogrutan Library Path Item Options.
8. Ändra länktypen till extern i dialogrutan Alternativ för objekt i bibliotekssökväg.

Nu kan du kompilera programmet med till exempel Project > Build Project.

Använda ANE-filen med Flash Professional

Om ditt program använder ett ANE-tillägg inkluderar du ANE-filen för tillägget i bibliotekssökvägen. Sedan kan du använda Flash Professional CS5.5 för att kompilera ActionScript-koden. Gör så här:

1. Ändra filnamnstillägget på ANE-filen från .ane till .swc. Detta steg är nödvändigt för att Flash Professional ska kunna hitta filen.
2. Välj Arkiv > ActionScript-inställningar för FLA-filen.
3. Välj fliken Bibliotekssökväg i dialogrutan Avancerade inställningar för ActionScript 3.0.
4. Klicka på knappen Bläddra efter SWC-fil.
5. Bläddra till SWC-filen och klicka på Öppna.

   SWC-filen visas nu på fliken Bibliotekssökväg i dialogrutan Avancerade inställningar för ActionScript 3.0.
6. Markera SWC-filen och klicka på knappen Ange länkningsalternativ för ett bibliotek.
7. Ändra länktypen till extern i dialogrutan Alternativ för objekt i bibliotekssökväg.

Paketera ett program som använder ANE-tillägg

Använd ADT för att paketera ett program som använder ANE-tillägg. Du kan inte paketera programmet med Flash Professional CS5.5 eller Flash Builder 4.5.1.

Mer information om ADT finns på AIR Developer Tool (ADT).

Exempelvis skapar följande ADT-kommandon en DMG-fil (en systemspecifik installationsfil för Mac OS X) för ett program som använder ANE-tillägg:

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

Följande kommando skapar ett APK-paket för en Android-enhet:

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

Följande kommando skapar ett iOS-paket för ett iPhone-program:

```
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
```

Tänk på följande:

- Använd en systemspecifik typ av installationspaket.
- Ange tilläggskatalogen.
- Kontrollera att ANE-filen har stöd för programmets målenhet.

Använd en systemspecifik typ av installationspaket

Programpaketet måste vara ett systemspecifikt installationsprogram. Du kan inte skapa ett AIR-paket för flera plattformar (ett .air-paket) för ett program där ANE-tillägg används, eftersom ANE-tillägg vanligtvis innehåller systemspecifik kod. Det är emellertid vanligt att samma ActionScript-API:er i ett ANE-tillägg kan användas för flera plattformar. I dessa fall kan du använda samma ANE-fil i olika systemspecifika installationspaket.

I följande tabell sammanställs de värden som kan användas för alternativet `-target` i ADT-kommandot:

Programmets målplattform	**-target**
Mac OS X- eller Windows-datorer	-target native -target bundle
Android	-target apk eller andra mål för Android-paket.
iOS	-target ipa-ad-hoc eller andra mål för iOS-paket
iOS-simulator	-target ipa-test-interpreter-simulator -target ipa-debug-interpreter-simulator

Ange tilläggskatalogen

Använd ADT-alternativet `-extdir` för att ange vilken katalog som innehåller ANE-filerna.

Mer information om detta alternativ finns i ”Alternativ för filer och sökvägar” på sidan 178.

Kontrollera att ANE-filen har stöd för programmets målenhet

När du ska tillhandahålla en ANE-fil får du av ANE-utvecklaren reda på vilka plattformar som tillägget har stöd för. Du kan även använda ett arkivextraheringsverktyg för att kontrollera innehållet i en ANE-fil. De extraherade filerna innehåller en katalog för varje plattform som stöds.

Att veta vilka plattformar som tillägget har stöd för är viktigt att känna till vid paketeringen av programmet där ANE-filen används. Tänk på följande regler:

- För att skapa ett Android-programpaket måste ANE-filen innehålla `Android-ARM`-plattformen. Alternativt måste ANE-filen innehålla standardplattformen samt minst ytterligare en plattform.
- För att skapa ett iOS-programpaket måste ANE-filen innehålla `iPhone-ARM`-plattformen. Alternativt måste ANE-filen innehålla standardplattformen samt minst ytterligare en plattform.
- För att skapa ett programpaket för iOS-simulatorn måste ANE-filen innehålla `iPhone-x86`-plattformen.
- För att skapa ett Mac OS X-programpaket måste ANE-filen innehålla `MacOS-x86`-plattformen. Alternativt måste ANE-filen innehålla standardplattformen samt minst ytterligare en plattform.
- För att skapa ett Windows-programpaket måste ANE-filen innehålla `Windows-x86`-plattformen. Alternativt måste ANE-filen innehålla standardplattformen samt minst ytterligare en plattform.

Kapitel 10: ActionScript-kompilerare

Innan ActionScript- och MXML-kod kan inkluderas i ett AIR-program måste den kompileras. Om du använder en integrerad utvecklingsmiljö, t.ex. Adobe Flash Builder eller Adobe Flash Professional, hanteras kompileringen i bakgrunden. Men du kan även anropa ActionScript-kompilerare från kommandoraden för att skapa SWF-filer om du inte använder en integrerad utvecklingsmiljö eller om du använder ett byggskript.

Om AIR-kommandoradsverktygen i Flex SDK

Varje kommandoradsverktyg som du använder för att skapa ett Adobe AIR-program anropar motsvarande verktyg som används för att skapa program:

- amxmlc anropar mxmlc för att kompilera programklasser
- acompc anropar compc för att kompilera biblioteks- och komponentklasser
- aasdoc anropar asdoc för att generera dokumentationsfiler från källkodskommentarer

Den enda skillnaden mellan Flex- och AIR-versionerna av verktygen är att AIR-versionerna läser in konfigurationsalternativen från filen air-config.xml i stället för flex-config.xml.

Flex SDK-verktygen och deras kommnadoradsalternativ beskrivs till fullo i Flex-dokumentationen. Flex SDK-verktygen beskrivs kortfattat här för att hjälpa dig att komma igång och för att visa skillnaderna mellan att bygga Flex-program och AIR-program.

Fler hjälpavsnitt

”Skapa ditt första AIR-skrivbordsprogram med Flex SDK” på sidan 36

Kompilatorinställningar

Du kan vanligtvis ange kompileringsalternativ både på kommandoraden och med en eller flera konfigurationsfiler. Den globala Flex SDK konfigurationsfilen innehåller standardvärden som används när kompilatorerna körs. Du kan redigera filen så att den passar din utvecklingsmiljö. Det finns två globala Flex konfigurationsfiler i katalogen frameworks i Flex SDK-installationen. Filen air-config.xml används när du kör kompilatorn amxmlc. Filen konfigurerar kompilatorn för AIR genom att inkludera AIR-biblioteken. Filen flex-config.xml används när du kör mxmlc.

Konfigurationens standardvärden lämpliga för att upptäcka hur Flex och AIR fungerar, men när du startar ett stort projekt bör du kontrollera de tillgängliga alternativen i detalj. Du kan ange projektspecifika värden för kompilatoralternativ i en lokal konfigurationsfil som har prioritet över de globala värdena för ett visst projekt.

Obs! *Inga kompileringsalternativ används speciellt för AIR-program men du måste skapa en referens till AIR-biblioteken när ett AIR-program kompileras. En referens till dessa bibliotek skapas i vanliga fall i en konfigurationsfil på projektnivå, i en fil för ett konstruktionsverktyg som Ant eller direkt på kommandoraden.*

Kompilera MXML- och ActionScript-källfiler för AIR

Du kan kompilera Adobe® ActionScript® 3.0- och MXML-resurserna i ditt AIR-program med kommandoradskompilatorn för MXML (amxmlc). (Du behöver inte kompilera HTML-baserade program. Om du vill kompilera en SWF-film i Flash Professional behöver du enbart publicera filmen som en SWF-fil.)

Det grundläggande kommandoradsmönstret för att använda amxmlc är:

```
amxmlc [compiler options] -- MyAIRApp.mxml
```

där *[kompilatoralternativ]* anger vilka kommandoradsalternativ som används för att kompilera AIR-programmet.

Kommandot amxmlc anropar den förvalda mxmlc-kompilatorn i Flex med en extra parameter, `+configname=air`. Den här parametern anger för kompilatorn att använda filen air-config.xml i stället för filen flex-config.xml. Annars använder du amxmlc på samma sätt som mxmlc.

Kompilatorn läser in konfigurationsfilen air-config.xml som anger vilka AIR- och Flex-bibliotek som vanligtvis krävs för att kompilera ett AIR-program. Du kan också läsa in en lokal konfigurationsfil på projektnivå om du vill åsidosätta eller tillföra tilläggsalternativ i den globala konfigurationen. Det enklaste sättet att skapa en lokal konfigurationsfil brukar vara att redigera en kopia av den globala versionen. Du kan läsa in den lokala filen med alternativet `-load-config`:

-load-config=project-config.xml Åsidosätter globala alternativ.

-load-config+=project-config.xml Lägger till flera värden i de globala alternativ som kan använda flera värden, till exempel alternativet `-library-path`. Globala alternativ som bara kan använda ett värde åsidosätts.

Om du använder en särskild namngivningsregel för den lokala konfigurationsfilen, läser amxmlc-kompilatorn in den lokala filen automatiskt. Om till exempel huvud-MXML-filen är `RunningMan.mxml` ska du namnge den lokala konfigurationsfilen så här: `RunningMan-config.xml`. För att kompilera programmet behöver du nu bara skriva:

```
amxmlc RunningMan.mxml
```

`RunningMan-config.xml` läses in automatiskt eftersom filnamnet matchar den kompilerade MXML-filen.

amxmlc, exempel

I följande exempel visas hur du använder amxmlc-kompilatorn. (Det är bara ActionScript- och MXML-resurserna i programmet som måste kompileras.)

Kompilera en AIR MXML-fil:

```
amxmlc myApp.mxml
```

Kompilera och ange utdatanamn:

```
amxmlc –output anApp.swf -- myApp.mxml
```

Kompilera en AIR ActionScript-fil:

```
amxmlc myApp.as
```

Ange kompilatorns konfigurationsfil:

```
amxmlc –load-config config.xml -- myApp.mxml
```

Lägga till fler alternativ från en annan konfigurationsfil:

```
amxmlc –load-config+=moreConfig.xml -- myApp.mxml
```

Lägga till bibliotek på kommandoraden (förutom biblioteken som redan finns i konfigurationsfilen):

```
amxmlc –library-path+=/libs/libOne.swc,/libs/libTwo.swc  -- myApp.mxml
```

Kompilera en AIR MXML-fil utan att använda en konfigurationsfil (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
```

Kompilera en AIR MXML-fil utan att använda en konfigurationsfil (Mac OS X eller 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
```

Kompilera en AIR MXML-fil till att använda ett bibliotek som delas under körning:

```
amxmlc -external-library-path+=../lib/myLib.swc -runtime-shared-libraries=myrsl.swf --
myApp.mxml
```

Kompilera en AIR MXML-fil så att den använder ett ANE-tillägg (se till att använda `-external-library-path` för ANE-tillägget):

```
amxmlc -external-library-path+=../lib/myANE.ane -output=myAneApp.swf -- myAneApp.mxml
```

Kompilera från Java (med klassökvägen inställd på att inkludera `mxmlc.jar`):

```
java flex2.tools.Compiler +flexlib [Flex SDK 3]/frameworks +configname=air [additional
compiler options] -- myApp.mxml
```

Alternativet flexlib identifierar var Flex SDK-ramverkskatalogen finns, så att kompilatorn kan hitta filen flex_config.xml.

Kompilera från Java (utan att klassökvägen har ställts in):

```
java -jar [Flex SDK 2]/lib/mxmlc.jar +flexlib [Flex SDK 3]/frameworks +configname=air
[additional compiler options] -- myApp.mxml
```

Så här anropas kompilatorn med Apache Ant (i exemplet används en Java-aktivitet för att köra 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>
```

Kompilera en AIR-komponent eller ett kodbibliotek (Flex)

Använd komponentkompilatorn, acompc, när du vill kompilera AIR-bibliotek och oberoende komponenter. Komponentkompilatorn acompc fungerar som amxmlc-kompilatorn, men med följande undantag:

- Du måste ange vilka klasser i kodbasen som ska tas med i biblioteket eller komponenten.
- acompc söker inte automatiskt efter en lokal konfigurationsfil. Om du vill använda projektkonfigurationsfilen måste du använda alternativet –load-config.

Kommandot acompc anropar den förvalda Flex-komponentkompilatorn compc, men läser in dess konfigurationsalternativ från filen `air-config.xml` i stället för filen `flex-config.xml`.

Komponentkompilatorns konfigurationsfil

Använd en lokal konfigurationsfil så att du inte behöver skriva (och eventuellt skriva fel) källsökvägen och klassnamnet på kommandoraden. Lägg till alternativet -load-config på acompc-kommandoraden om du vill läsa in den lokala konfigurationsfilen.

I följande exempel visas en konfiguration som bygger ett bibliotek med två klasser, ParticleManager och Particle, där båda finns i paketet: `com.adobe.samples.particles`. Klassfilerna finns i mappen `source/com/adobe/samples/particles`.

```
<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>
```

Du kompilerar biblioteket med hjälp av konfigurationsfilen med namnet `ParticleLib-config.xml` genom att skriva:

```
acompc -load-config ParticleLib-config.xml -output ParticleLib.swc
```

Du kör samma kommando helt på kommandoraden genom att skriva:

```
acompc -source-path source -include-classes com.adobe.samples.particles.Particle
com.adobe.samples.particles.ParticleManager -output ParticleLib.swc
```

(Skriv hela kommandot på raden, eller använd fortsättningstecknet för kommandoskalet.)

acompc, exempel

I dessa exempel förutsätts det att du använder en konfigurationsfil med namnet `myLib-config.xml`.

Kompilera en AIR-komponent eller ett -bibliotek:

```
acompc -load-config myLib-config.xml -output lib/myLib.swc
```

Kompilera ett bibliotek som delas under körning:

```
acompc -load-config myLib-config.xml -directory -output lib
```

(Mappen lib måste finnas och vara tom innan du kör kommandot.)

Kapitel 11: AIR Debug Launcher (ADL)

Använd AIR Debug Launcher (ADL) om du vill köra både SWF-baserade och HTML-baserade program under utveckling. Med ADL kan du köra ett program utan att först förpacka och installera det. ADL använder som standard en körningsversion som ingår i SDK, vilket innebär att du inte behöver installera körningsversionen separat för att kunna använda ADL.

ADL skriver ut trace-programsatser och körningsfelen i standardutdata, men har inte stöd för brytpunkter eller andra felsökningsfunktioner. Du kan använda Flash Debugger (eller en integrerad utvecklingsmiljö, till exempel Flash Builder) för komplexa felsökningsproblem.

Obs! *Om* `trace()` *-programsatserna inte visas på konsolen kontrollerar du att du inte har angett* `ErrorReportingEnable` *eller* `TraceOutputFileEnable` *i filen mm.cfg. Du hittar mer information om var den här filen finns på olika plattformar i Redigera filen mm.cfg.*

AIR har direkt stöd för felsökning vilket innebär att du inte behöver en felsökningsversion av körtidsmodulen (till skillnad från Adobe® Flash® Player). Använd Flash Debugger och AIR Debug Launcher (ADL) när du ska utföra felsökning på kommandoraden.

Flash Debugger distribueras i katalogen Flex SDK. Interna versioner, t.ex. fdb.exe för Windows, finns i underkatalogen bin. Java-versionen finns i underkatalogen lib. AIR Debug Launcher, adl.exe, finns i katalogen bin för Flex SDK-installationen. (Det finns inte en separat Java-version).

Obs! *Du kan inte starta ett AIR-program direkt med fdb, eftersom fdb försöker starta det med Flash Player. Låt i stället AIR-programmet ansluta till en fdb-session som körs.*

ADL-användning

Använd följande mönster när du kör ett program med ADL:

```
adl application.xml
```

Här är *application.xml* programbeskrivningsfilen för det aktuella programmet.

Den fullständiga syntaxen för ADL är:

```
adl [-runtime runtime-directory]
    [-pubid publisher-id]
    [-nodebug]
    [-atlogin]
    [-profile profileName]
    [-screensize value]
    [-extdir extension-directory]
    application.xml
    [root-directory]
    [-- arguments]
```

(Objekt inom hakparenteser, [], är valfria.)

-runtime *körningsbibliotek* Anger katalogen som innehåller den körningsmiljö som ska användas. Om ingen anges används körningskatalogen som finns i samma SDK som ADL-programmet. Du måste ange körningskatalogen om du flyttar ADL bort från dess SDK-mapp. I Windows och Linux anger du katalogen som innehåller `Adobe AIR`-katalogen. I Mac OS X anger du katalogen som innehåller `Adobe AIR.framework`.

***-pubid publisher-id* Tilldelar det angivna värdet som utgivar-ID för AIR-programmet för den här körningen.** Om du anger ett tillfälligt utgivar-ID kan du testa funktionerna i ett AIR-program, till exempel kommunicera via en lokal anslutning, som använder utgivar-ID:t för att unikt identifiera ett program. Från och med AIR 1.5.3 kan du även ange utgivar-ID i programbeskrivningsfilen (och bör inte använda den här parametern).

***Obs!** Från och med AIR 1.5.3 beräknas och tilldelas inte längre ett utgivar-ID automatiskt till ett AIR-program. Du kan ange ett utgivar-ID när du skapar en uppdatering till ett befintligt AIR-program, men nya program behöver inte och bör inte ange ett utgivar-ID.*

-nodebug Stänger av felsökningsstödet. Om detta används kan programprocessen inte ansluta till Flash-felsökaren och dialogrutor för ohanterade undantag undertrycks. (Trace-programsatserna skrivs emellertid fortfarande till konsolfönstret.) Om du stänger av felsökningen kan programmet köra lite snabbare och också bättre emulera körningsläget för ett installerat program.

-atlogin Simulerar programstart vid inloggning. Med den här flaggan kan du testa programlogik som bara körs när ett program är inställt på att startas när användaren loggar in. När du använder `-atlogin` kommer den `reason`-egenskap för InvokeEvent-objektet som skickas till programmet att vara `login` i stället för `standard` (om inte programmet redan är igång).

***-profile profileName* Den angivna profilen används i ADL för att felsöka programmet.** *`profileName`* kan vara något av följande värden:

- desktop
- extendedDesktop
- mobileDevice

Om programbeskrivningen innehåller ett `supportedProfiles`-element måste den profil du anger med `-profile` vara medlem i den lista som stöds. Om du inte använder flaggan `-profile` används den första profilen i programbeskrivningsfilen som aktiv profil. Om programbeskrivningen inte innehåller `supportedProfiles`-elementet, och du inte använder flaggan `-profile`, används profilen *desktop*.

Du hittar mer information i avsnitten ”supportedProfiles” på sidan 234 och ”Enhetsprofiler” på sidan 241.

-screensize *värde* Den simulerade skärmstorlek som ska användas när program körs i profilen mobileDevice på skrivbordet. Ange skärmstorleken som en fördefinierad skärmtyp eller som den normala breddens och höjdens pixelmått för stående layout, plus helskärmsbildens bredd och höjd. Använd en av följande fördefinierade skärmtyper för att ange värdet efter typ:

Skärmtyp	Normal bredd x höjd	Helskärm bredd x höjd
480	720 x 480	720 x 480
720	1 280 x 720	1 280 x 720
1 080	1 920 x 1 080	1 920 x 1 080
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 1 004	768 x 1 024
iPadRetina	1 536 x 2 008	1 536 x 2 048
iPhone	320 x 460	320 x 480

Skärmtyp	Normal bredd x höjd	Helskärm bredd x höjd
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 x 1096	640 x 1136
NexusOne	480 x 762	480 x 800
QVGA	240 x 320	240 x 320
SamsungGalaxyS	480 x 762	480 x 800
SamsungGalaxyTab	600 x 986	600 x 1 024
WQVGA	240 x 400	240 x 400
WVGA	480 x 800	480 x 800

Använd följande format om du vill ange skärmens pixelmått direkt:

```
widthXheight:fullscreenWidthXfullscreenHeight
```

Ange alltid pixeldimensioner för stående layout, d.v.s. ange bredden som ett värde som är mindre än höjden. Du kan till exempel ange NexusOne-skärmen med:

```
-screensize 480x762:480x800
```

-extdir *extension-directory* Den katalog i vilken körningsmiljön ska söka efter ANE-tillägg. Katalogen innehåller en underkatalog för varje ANE-tillägg som används i programmet. Var och en av dessa underkataloger innehåller den *opaketerade* ANE-filen för ett tillägg. Till exempel:

```
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
```

Tänk på följande när du använder parametern -extdir:

- För ADL-kommandot krävs att varje angiven katalog har filnamnstillägget .ane. Filnamnsdelen före suffixet ".ane" kan emellertid vara ett valfritt godkänt filnamn. Det behöver *inte* matcha värdet för elementet `extensionID` i programbeskrivningsfilen.
- Du kan ange parametern `-extdir` flera gånger.
- Användningen av parametern `-extdir` skiljer sig åt mellan verktygen ADT och ADL. I ADT anger parametern en katalog som innehåller ANE-filer.
- Du kan även använda miljövariabeln `AIR_EXTENSION_PATH` för att ange tilläggskatalogen. Läs mer i ”ADT-systemvariabler” på sidan 184.

application.xml Programbeskrivningsfilen. Läs mer i ”AIR-programbeskrivningsfiler” på sidan 200. Programbeskrivningsfilen är den enda parametern som krävs för ADL och i de flesta fall även den enda parametern som krävs.

root-directory Anger rotkatalogen för programmet som ska köras. Om ingen anges används katalogen som innehåller programbeskrivningsfilen.

-- ***arguments*** Den teckensträng som visas efter "--" skickas till programmet som kommandoradsargument.

Obs! *När du startar ett AIR-program som redan körs startas inte en ny instans av det programmet. I stället skickas en* `invoke`*-händelse till den instans som körs.*

ADL-exempel

Köra ett program i aktuell katalog:

```
adl myApp-app.xml
```

Köra ett program i en underkatalog till den aktuella katalogen:

```
adl source/myApp-app.xml release
```

Kör ett program och skicka två kommandoradsargument, "tick" och "tock":

```
adl myApp-app.xml -- tick tock
```

Köra ett program med hjälp av en viss körningsversion:

```
adl -runtime /AIRSDK/runtime myApp-app.xml
```

Köra ett program utan felsökningsstöd:

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

Köra ett program med profilen för mobilenheter och simulera NexusOne-skärmstorlek:

```
adl -profile mobileDevice -screensize NexusOne myMobileApp-app.xml
```

Köra ett program med Apache Ant (sökvägarna i exemplet gäller 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>
```

Avslutnings- och felkoder för ADL

I tabellen nedan beskrivs de avslutningskoder som skrivs ut av ADL:

Avslutningskod	Beskrivning
0	Lyckad start. ADL avslutas när AIR-programmet avslutas.
1	Lyckat anrop av ett AIR-program som redan körs. ADL avslutas omedelbart.
2	Användningsfel. Argumenten som anges för ADL är felaktiga.
3	Körningsversionen kan inte hittas.
4	Körningsversionen kan inte startas. Detta sker ofta på grund av att den version som anges i programmet inte matchar körningsmiljöns version.
5	Ett okänt fel har inträffat.
6	Programbeskrivningsfilen kan inte hittas.
7	Innehållet i programbeskrivningsfilen är inte giltigt. Det här felet anger vanligtvis att XML är felformaterat.
8	Huvudprogrammets innehållsfil (anges i elementet <content> för programbeskrivningsfilen) kan inte hittas.

Avslutningskod	Beskrivning
9	Huvudprogrammets innehållsfil är inte en giltig SWF- eller HTML-fil.
10	Programmet saknar stöd för den profil som angetts med alternativet -profile.
11	Argumentet -screensize stöds inte i den aktuella profilen.

Kapitel 12: AIR Developer Tool (ADT)

ADT (AIR Developer Tool) är ett kommandoradsverktyg för att utveckla AIR-program. Du kan använda ADT för att utföra följande:

- Paketera ett AIR-program som en .air-installationsfil
- Paketera ett AIR-program som ett systemspecifikt installationsprogram, till exempel som en .exe-fil på Windows, en .ipa-fil på iOS eller en .apk-fil på Android
- Paketera ett ANE-tillägg som en ANE-fil (AIR Native Extension)
- Signera ett AIR-program med ett digitalt certifikat
- Ändra (flytta) den digitala signatur som används för programuppdateringar
- Avgöra vilka enheter som är anslutna till en dator
- Skapa ett självsignerat digitalt kodsigneringscertifikat
- Fjärrinstallera, fjärrstarta och fjärravinstallera ett program på en mobilenhet
- Fjärrinstallera och fjärravinstallera AIR-miljön på en mobilenhet

ADT är ett Java-program som ingår i AIR SDK. Du måste ha Java 1.5 eller senare för att använda det. SDK innehåller en skriptfil för att anropa ADT. Om du vill använda det här skriptet måste sökvägen till Java-programmet inkluderas i systemvariabeln path. Om `bin`-katalogen i AIR SDK även anges i systemvariabeln path kan du skriva `adt` på kommandoraden, med lämpliga argument, för att anropa ADT. (Om du inte vet hur du anger systemvariabeln path läser du dokumentationen till operativsystemet. Anvisningar om hur du anger sökvägen (path) på de flesta system beskrivs i avsnittet ”Systemvariabeln path” på sidan 299.)

Det krävs minst 2 GB minne för att använda ADT. Om du inte har så mycket minne kan ADT få slut på minne, särskilt när program för iOS paketeras.

Förutsatt att bin-katalogen för både Java och AIR SDK ingår i variabeln path kan du köra ADT med följande grundläggande syntax:

```
adt -command options
```

Obs! *I de flesta integrerade utvecklingsmiljöer, inklusive Adobe Flash Builder och Adobe Flash Professional, kan du paketera och signera AIR-program. Du behöver vanligtvis inte använda ADT för dessa vanliga uppgifter när du redan arbetar i en sådan utvecklingsmiljö. Du kan dock behöva använda ADT som kommandoradsverktyg för funktioner som inte stöds i den integrerade utvecklingsmiljön. Dessutom kan du använda ADT som kommandoradsverktyg som en del i en automatiserad byggprocess.*

ADT-kommandon

Det första argument som skickas till ADT anger ett av följande kommandon.

- -package – Paketerar ett AIR-program eller en ANE-fil (AIR Native Extension).
- -prepare – Paketerar ett AIR-program som en AIRI-fil (AIR Intermediate), men signerar den inte. Det går inte att installera AIRI-filer.

- -sign – Signerar ett AIRI-paket som skapats med kommandot -prepare. Med kommandona -prepare och -sign kan paketering och signering utföras vid olika tidpunkter. Du kan också använda kommandot -sign för att signera eller signera om ett ANE-paket.
- -migrate – Använder en flyttningssignatur på ett signerat AIR-paket, så att du kan använda ett nytt eller förnyat kodsigneringscertifikat.
- -certificate – Skapar ett självsignerat digitalt kodsigneringscertifikat.
- -checkstore – Verifierar att det går att komma åt ett digitalt certifikat i en nyckelbehållare.
- -installApp – Installerar ett AIR-program på en enhet eller en enhetsemulator.
- -launchApp – Startar ett AIR-program på en enhet eller en enhetsemulator.
- -appVersion – Rapporterar vilken version av ett AIR-program som för tillfället är installerad på en enhet eller en enhetsemulator.
- -uninstallApp – Avinstallerar ett AIR-program från en enhet eller en enhetsemulator.
- -installRuntime – Installerar AIR-miljön på en enhet eller en enhetsemulator.
- -runtimeVersion – Rapporterar vilken version av AIR-miljön som för tillfället är installerad på en enhet eller en enhetsemulator.
- -uninstallRuntime – Avinstallerar den installerade AIR-miljön från en enhet eller en enhetsemulator.
- -version – Rapporterar versionsnumret på ADT.
- -devices – Rapporterar enhetsinformation för anslutna mobilenheter och emulatorer.
- -help – Visar en lista över kommandon och alternativ.

Många ADT-kommandon delar relaterade uppsättningar med alternativflaggor och parametrar. De här uppsättningarna med alternativ beskrivs separat:

- ”ADT-kodsigneringsalternativ” på sidan 176
- ”Alternativ för filer och sökvägar” på sidan 178
- ”Anslutningsalternativ för felsökning” på sidan 179
- ”Alternativ för ANE (Native Extension)” på sidan 180

ADT-kommandot package

Kommandot `-package` bör köras från huvudprogramkatalogen. Kommandot har följande syntaxer:

Skapa ett AIR-paket från komponentens programfiler:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    -sampler
    -hideAneLibSymbols
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    FILE_OPTIONS
```

Skapa ett systemspecifikt paket från komponentens programfiler:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target packageType
    DEBUGGER_CONNECTION_OPTIONS
    -airDownloadURL URL
    NATIVE_SIGNING_OPTIONS
    output
    app_descriptor
    -platformsdk path
    FILE_OPTIONS
```

Skapa ett systemspecifikt paket som innehåller ett systemspecifikt tillägg från komponentens programfiler:

```
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
```

Skapa ett systemspecifikt paket från en AIR- eller AIRI-fil:

```
adt -package
    -target packageType
    NATIVE_SIGNING_OPTIONS
    output
    input_package
```

Skapa ett systemspecifikt tilläggspaket av komponentens ANE-tilläggsfiler:

```
adt -package
    AIR_SIGNING_OPTIONS
    -target ane
    output
    ANE_OPTIONS
```

Obs! *Du behöver inte signera en ANE-fil, så parametern* `AIR_SIGNING_OPTIONS` *är valfri i detta exempel.*

AIR_SIGNING_OPTIONS Signeringsalternativen för AIR identifierar det certifikat som används för att signera en AIR-installationsfil. Signeringsalternativen beskrivs ingående i ”ADT-kodsigneringsalternativ” på sidan 176.

-migrate Den här flaggan anger att programmet signeras med ett flyttningscertifikat utöver det certifikat som anges i parametrarna `AIR_SIGNING_OPTIONS`. Den här flaggan är bara giltig om du paketerar ett skrivbordsprogram som ett systemspecifikt installationsprogram och programmet använder ett systemspecifikt tillägg. I andra fall genereras ett fel. Signeringsalternativen för flyttningscertifikatet anges som **MIGRATION_SIGNING_OPTIONS**-parametrar. Dessa signeringsalternativ beskrivs ingående i ”ADT-kodsigneringsalternativ” på sidan 176. Med flaggan `-migrate` kan du skapa en uppdatering för ett systemspecifikt installationsprogram för ett skrivbordsprogram som använder ett systemspecifikt tillägg och ändra programmets kodsigneringscertifikat, som när det ursprungliga certifikatet upphör att gälla. Du hittar mer information i ”Signera en uppdaterad version av ett AIR-program” på sidan 195.

Flaggan `-migrate` i kommandot `-package` finns i AIR 3.6 och senare versioner.

-target Den typ av paket som ska skapas. De pakettyper som stöds är:

- air – Ett AIR-paket. ”air” är standardvärdet, och flaggan -target behöver inte anges när du skapar AIR- eller AIRI-filer.
- airn – Ett systemspecifikt programpaket för enheter i den utökade tv-profilen.
- ane – Ett ANE-tilläggspaket (AIR Native Extension).
- Mål för Android-paket:
  - apk – Ett Android-paket. Ett paket som skapas med det här målet kan bara installeras på en Android-enhet, inte på en emulator.
  - apk-captive-runtime – ett Android-paket som innehåller både programmet och en låst version AIR-miljön. Ett paket som skapas med det här målet kan bara installeras på en Android-enhet, inte på en emulator.
  - apk-debug – Ett Android-paket med extra felsökningsinformation. (Även SWF-filerna i programmet måste kompileras med felsökningsstöd.)
  - apk-emulator – Ett Android-paket för användning på en emulator utan felsökningsstöd. (Använd målet apk-debug om du vill tillåta felsökning på både emulatorer och enheter.)
  - apk-profile – Ett Android-paket med stöd för programprestanda och minnesprofilering.
- Mål för iOS-paket:
  - ipa-ad-hoc – Ett iOS-paket för ad hoc-distribution.
  - ipa-app-store – Ett iOS-paket för Apple App Store-distribution.
  - ipa-debug – Ett iOS-paket med extra felsökningsinformation. (Även SWF-filerna i programmet måste kompileras med felsökningsstöd.)
  - ipa-test – Ett iOS-paket som kompilerats utan optimering eller felsökningsinformation.
  - ipa-debug-interpreter – funktionell motsvarighet till ett felsökningspaket, men med snabbare kompilering. Dock kommer ActionScript-bytekod att tolkas och inte översättas till maskinkod. Detta medför att kodkörningen är långsammare i ett tolkat paket.
  - ipa-debug-interpreter-simulator – funktionell motsvarighet till ipa-debug-interpreter, men förpackad för iOS-simulatorn. Endast Macintosh. Om du använder detta alternativ måste du även inkludera alternativet -platformsdk genom att ange sökvägen till SDK:n för iOS-simulatorn.
  - ipa-test-interpreter – funktionell motsvarighet till ett testpaket, men med snabbare kompilering. Dock kommer ActionScript-bytekod att tolkas och inte översättas till maskinkod. Detta medför att kodkörningen är långsammare i ett tolkat paket.
  - ipa-test-interpreter-simulator – funktionell motsvarighet till ipa-test-interpreter, men förpackad för iOS-simulatorn. Endast Macintosh. Om du använder detta alternativ måste du även inkludera alternativet -platformsdk genom att ange sökvägen till SDK:n för iOS-simulatorn.
- native – Ett systemspecifikt installationsprogram för stationära datorer. Vilken typ av fil som skapas beror på vilket operativsystem kommandot körs på (filen får systemets installationsformat):
  - EXE – Windows
  - DMG – Mac
  - DEB – Ubuntu Linux (AIR 2.6 eller tidigare)
  - RPM – Fedora eller OpenSuse Linux (AIR 2.6 eller senare)

  Du hittar mer information i ”Paketera ett systemspecifikt installationsprogram för skrivbordet” på sidan 55.

-sampler (endast iOS, AIR 3.4 eller senare) Aktiverar den telemetribaserade ActionScript-samplern i iOS-program. Om du använder denna flagga kan du profilera programmet med Adobe Scout. Även om Scout kan profilera allt innehåll på Flash-plattformar kan du med telemetri få bättre inblick i hur anpassning, DisplayList, Stage3D-återgivning och annat fungerar i ActionScript-funktioner. Observera att denna flagga kommer att påverka prestandan något, så använd den inte i produktionsfärdiga program.

-hideAneLibSymbols (endast iOS, AIR 3.4 och senare) Programutvecklare kan använda flera ANE-tillägg från flera källor och om ANE-tilläggen delar ett gemensamt symbolnamn, kommer ADT att generera felet "duplicerad symbol i objektfil". I vissa fall kan detta fel manifesteras som en krasch under körningen. Du kan använda alternativet `hideAneLibSymbols` för att göra eller inte göra ANE-bibliotektetsymboler synliga endast för det bibliotekets källor (yes) eller synliga globalt (no):

- **yes** – Döljer ANE-symboler vilket löser eventuella problem med oönskade symbolkonflikter.
- **no** – (standard) ANE-symboler döljs inte. Detta var vad som gällde före AIR 3.4.

-embedBitcode (endast iOS, AIR 25 och senare) Programutvecklare kan använda alternativet `embedBitcode` för att ange om bitkod ska bäddas in i iOS-programmet eller inte genom att ange detta som yes eller no. Standardvärdet för den här växeln är no om inget annat anges.

DEBUGGER_CONNECTION_OPTIONS Med anslutningsalternativen för felsökning anger du om ett felsökningspaket ska försöka ansluta till en fjärrfelsökare som körs på en annan dator eller lyssna efter en anslutning från en fjärrfelsökare. De här alternativen stöds bara för mobilfelsökningspaket (med målen apk-debug och ipa-debug). Alternativen beskrivs i ”Anslutningsalternativ för felsökning” på sidan 179.

-airDownloadURL Anger en alternativ webbadress för att hämta och installera AIR-miljön på Android-enheter. Om detta inte anges dirigeras användaren om till AIR-miljön på Android Market om miljön inte redan har installerats.

Om ditt program distribueras via en annan plats (annan än Android Market, som administreras av Google) kanske du måste ange webbadressen för hämtning av AIR-miljön från den platsen. På vissa andra platser tillåts inte att program kräver hämtning från externa platser. Det här alternativet stöds bara för Android-paket.

NATIVE_SIGNING_OPTIONS De systemspecifika signeringsalternativen identifierar det certifikat som används för att signera en systemspecifik paketfil. De här signeringsalternativen används för att tillämpa en signatur som används av det inbyggda operativsystemet, inte av AIR-miljön. Alternativen är i övrigt identiska med AIR_SIGNING_OPTIONS och beskrivs detaljerat i ”ADT-kodsigneringsalternativ” på sidan 176.

Systemspecifika signaturer stöds på Windows och Android. På Windows bör både signeringsalternativen för AIR och de systemspecifika signeringsalternativen anges. På Android kan bara systemspecifika signeringsalternativ anges.

Ofta kan du använda samma kodsigneringscertifikat för både en AIR-signatur och en systemspecifik signatur. Det finns emellertid undantag. Till exempel anger Googles policy för program som skickas till Android Market att alla program måste signeras med ett certifikat som är giltigt till minst år 2033. Det innebär att ett certifikat som utfärdats av en välkänd certifikatutfärdare, vilket rekommenderas för AIR-signaturer, inte bör användas för att signera ett program för Android. (Det finns inga certifikatutfärdare som utfärdar kodsigneringscertifikat med så lång giltighet.)

output Namnet på den paketfil som ska skapas. Du kan ange filtillägget eller inte, det är valfritt. Om du inte anger det läggs ett tillägg som passar värdet på -target och det aktuella operativsystemet till.

app_descriptor Sökvägen till programbeskrivningsfilen. Sökvägen kan anges i förhållande till aktuell katalog eller som en absolut sökväg. (Programbeskrivningsfilen byter namn till *application.xml* i AIR-filen.)

-platformsdk Sökvägen till plattformens SDK för målenheten:

- Android – AIR 2.6+ SDK innehåller de verktyg från Android SDK som behövs för att implementera relevanta ADT-kommandon. Ange bara det här värdet om du vill använda en annan version av Android SDK. Om systemvariabeln AIR_ANDROID_SDK_HOME redan har angetts behöver sökvägen till plattforms-SDK inte anges på kommandoraden. (Om båda anges används den sökväg som anges på kommandoraden.)
- iOS – AIR SDK levereras med ett låst iOS SDK. Alternativet -platformsdk använder du för att förpacka program med en extern SDK så att du inte blir begränsad till att använda det iOS SDK som är låst. Om du till exempel har skapat ett tillägg med den senaste iOS SDK:n kan du ange detta SDK när du paketerar programmet. Dessutom när du använder ADT med iOS-simulatorn måste du alltid inkludera alternativet -platformsdk för att ange sökvägen till SDK:n för iOS-simulatorn.

-arch Programutvecklare kan använda detta argument för att skapa en APK för x86-plattformar och följande värden kan användas:

- armv7 - ADT förpackar APK för Android armv7-plattformen.
- x86 - ADT förpackar APK för Android x86-plattformen.

armv7 är standard när inget värde anges

FILE_OPTIONS Identifierar de programfiler som ska inkluderas i paketet. Filalternativen beskrivs mer ingående i ”Alternativ för filer och sökvägar” på sidan 178. Ange inte filalternativ när du skapar systemspecifika paket från en AIR- eller AIRI-fil.

input_airi Ange detta när du skapar ett systemspecifikt paket från en AIRI-fil. Du måste ange AIR_SIGNING_OPTIONS om målet är *air* (eller om inget mål anges).

input_air Ange detta när du skapar ett systemspecifikt paket från en AIR-fil. Ange inte AIR_SIGNING_OPTIONS.

ANE_OPTIONS Identifierar alternativ och filer för att skapa ett systemspecifikt tilläggspaket. Alternativen för tilläggspaketet beskrivs tillfullo i ”Alternativ för ANE (Native Extension)” på sidan 180.

Exempel på kommandon för ADT-paket

Paketspecifika programfiler i den aktuella katalogen för ett SWF-baserat AIR-program:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf components.swc
```

Paketspecifika programfiler i den aktuella katalogen för ett HTML-baserat AIR-program:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html AIRAliases.js
image.gif
```

Paketera alla filer och underkataloger i den aktuella arbetskatalogen:

```
adt –package -storetype pkcs12 -keystore ../cert.p12 myApp.air myApp.xml .
```

Obs! *Nyckelfilen som innehåller den privata nyckel som används för att signera programmet. Ta aldrig med signeringscertifikatet i AIR-paketet! Om du använder jokertecken i ADT-kommandot placerar du nyckelfilen på en annan plats så att den inte tas med i paketet. I det här exemplet finns nyckelfilen cert.p12 i den överordnade katalogen.*

Paketera bara huvudfilerna och en bildunderkatalog:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf images
```

Paketera ett HTML-baserat program och alla filer i HTML, skript och bildunderkataloger:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml index.html AIRALiases.js
html scripts images
```

Paketera programmets XML-fil och huvud-SWF-fil som finns i en arbetskatalog (release/bin):

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml –C
release/bin myApp.swf
```

Paketera resurser från mer än en plats i byggfilsystemet. I det här exemplet finns programresurserna i följande mappar före paketeringen:

```
/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
```

Om du kör följande ADT-kommando från katalogen /devRoot/myApp:

```
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
```

Leder detta till följande paketstruktur:

```
/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
```

Kör ADT som ett Java-program för ett enkelt SWF-baserat program (utan att ange klassökväg):

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.swf
```

Kör ADT som ett Java-program för ett enkelt HTML-baserat program (utan att ange klassökväg):

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.html AIRAliases.js
```

Köra ADT som ett Java-program (med Java classpath inställt på att inkludera paketet ADT.jar):

```
java -com.adobe.air.ADT –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml
myApp.swf
```

Kör ADT som ett Java-program i Apache Ant (trots att det vanligtvis är bäst att använda ADT-kommandot direkt i Ant-skriptet). Sökvägen som visas i exemplet gäller för 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>
```

Obs! *På vissa system kan dubbelbytetecken i filsystemsökvägar feltolkas. Om detta händer provar du med att ange teckenuppsättningen UTF-8 för den JRE-version som används för att köra ADT. Detta utförs som standard i det skript som används för att starta ADT på Mac och Linux. I Windows-filen adt.bat, eller om du kör ADT direkt från Java, anger du alternativet* `-Dfile.encoding=UTF-8` *på Java-kommandoraden.*

ADT-kommandot prepare

Kommandot -prepare skapar ett osignerat AIRI-paket. Ett AIRI-paket kan inte användas fristående. Använd kommandot -sign för att konvertera en AIRI-fil till ett signerat AIR-paket eller kommandot package för att konvertera AIRI-filen till ett systemspecifikt paket.

Kommandot -prepare har följande syntax:

```
adt -prepare output app_descriptor FILE_OPTIONS
```

output Namnet på den AIRI-fil som skapas.

app_descriptor Sökvägen till programbeskrivningsfilen. Sökvägen kan anges i förhållande till aktuell katalog eller som en absolut sökväg. (Programbeskrivningsfilen byter namn till *application.xml* i AIR-filen.)

FILE_OPTIONS Identifierar de programfiler som ska inkluderas i paketet. Filalternativen beskrivs mer ingående i ”Alternativ för filer och sökvägar” på sidan 178.

ADT-kommandot sign

Kommandot -sign signerar AIRI- och ANE-filer.

Kommandot -sign har följande syntax:

```
adt -sign AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS Signeringsalternativen för AIR identifierar det certifikat som används för att signera en paketfil. Signeringsalternativen beskrivs ingående i ”ADT-kodsigneringsalternativ” på sidan 176.

input Namnet på den AIRI- eller ANE-fil som ska signeras.

output Namnet på det signerade paket som ska skapas.

Om ANE-filen redan har signerats ignoreras den gamla signaturen. (Det går inte att signera om AIR-filer. Om du vill använda en ny signatur för en programuppdatering använder du kommandot `-migrate`.)

ADT-kommandot migrate

Kommandot -migrate använder en flyttningssignatur på en AIR-fil. Du måste använda en flyttningssignatur när du förnyar eller ändrar det digitala certifikatet och behöver uppdatera program som signerats med det gamla certifikatet.

Mer information om hur du paketerar AIR-program med en flyttningssignatur finns i ”Signera en uppdaterad version av ett AIR-program” på sidan 195.

Obs! *Flyttningscertifikatet måste användas inom 365 dagar från certifikatets förfallodatum. När den här fristen har gått ut kan dina programuppdateringar inte längre signeras med en flyttningssignatur. Användarna kan först uppdatera till en version av ditt program som signerats med en flyttningssignatur och sedan installera den senaste uppdateringen eller avinstallera det ursprungliga programmet och installera det nya AIR-paketet.*

Om du vill använda en flyttningssignatur signerar du först AIR-programmet med det nya eller förnyade certifikatet (med kommandot -package eller -sign) och använder sedan flyttningssignaturen tillsammans med det gamla certifikatet och kommandot -migrate.

Kommandot `-migrate` har följande syntax:

```
adt -migrate AIR_SIGNING_OPTIONS input output
```

AIR_SIGNING_OPTIONS Signeringsalternativen för AIR identifierar det ursprungliga certifikat som användes för att signera befintliga versioner av AIR-programmet. Signeringsalternativen beskrivs ingående i ”ADT-kodsigneringsalternativ” på sidan 176.

input Den AIR-fil som redan signerats med det NYA programcertifikatet.

output Namnet på det slutliga paketet med signaturer från både det nya och det gamla certifikatet.

Namnen på de filer som används för in- och utdata i AIR måste vara olika.

Obs! *ADT-kommandot migrate kan inte användas med AIR-program för datorer som inkluderar systemspecifika tillägg, eftersom sådana program paketeras som systemspecifika installationsprogram och inte som .air-filer. Om du vill ändra certifikat för ett AIR-program för datorer som inkluderar ett systemspecifikt tillägg paketerar du programmet med ”ADT-kommandot package” på sidan 163 och flaggan -migrate.*

ADT-kommandot checkstore

Med kommandot -checkstore kan du kontrollera giltigheten för en nyckelbehållare. Kommandot har följande syntax:

```
adt -checkstore SIGNING_OPTIONS
```

SIGNING_OPTIONS De signeringsalternativ som identifierar den nyckelbehållare som ska kontrolleras. Signeringsalternativen beskrivs ingående i ”ADT-kodsigneringsalternativ” på sidan 176.

ADT-kommandot certificate

Med kommandot -certificate kan du skapa ett självsignerat digitalt kodsigneringscertifikat. Kommandot har följande syntax:

```
adt -certificate -cn name -ou orgUnit -o orgName -c country -validityPeriod years key-type
output password
```

-cn Den sträng som tilldelas som det nya certifikatets vanliga namn.

-ou En sträng som tilldelas den organisationsenhet som utfärdar certifikatet. (Valfritt.)

`-o` En sträng som tilldelas den organisation som utfärdar certifikatet. (Valfritt.)

`-c` En ISO-3166-landskod på två bokstäver. Inget certifikat skapas om en ogiltig kod anges. (Valfritt.)

`-validityPeriod` Det antal år som certifikatet är giltigt. Om inget värde anges tilldelas en giltighet på fem år. (Valfritt.)

`key_type` Den typ av nyckel som ska användas för certifikatet är *2048-RSA*.

output Sökvägen till och filnamnet på den certifikatfil som ska skapas.

password Lösenordet för att komma åt det nya certifikatet. Lösenordet behövs när du signerar AIR-filer med det här certifikatet.

ADT-kommandot installApp

Kommandot -installApp installerar ett program på en enhet eller emulator.

Du måste avinstallera det befintliga programmet innan du kan installera om med det här kommandot.

Kommandot har följande syntax:

```
adt -installApp -platform platformName -platformsdk path-to-sdk -device deviceID -package
fileName
```

-platform Namnet på enhetens plattform. Ange *ios* eller *android*.

-platformsdk Sökvägen till plattformens SDK för målenheten (valfritt):

- Android – AIR 2.6+ SDK innehåller de verktyg från Android SDK som behövs för att implementera relevanta ADT-kommandon. Ange bara det här värdet om du vill använda en annan version av Android SDK. Om systemvariabeln AIR_ANDROID_SDK_HOME redan har angetts behöver sökvägen till plattforms-SDK inte anges på kommandoraden. (Om båda anges används den sökväg som anges på kommandoraden.)
- iOS – AIR SDK levereras med ett låst iOS SDK. Alternativet -platformsdk använder du för att förpacka program med en extern SDK så att du inte blir begränsad till att använda det iOS SDK som är låst. Om du till exempel har skapat ett tillägg med den senaste iOS SDK:n kan du ange detta SDK när du paketerar programmet. Dessutom när du använder ADT med iOS-simulatorn måste du alltid inkludera alternativet -platformsdk för att ange sökvägen till SDK:n för iOS-simulatorn.

-device Ange *ios_simulator*, serienummer (Android) eller referens (handle) (iOS) för den anslutna enheten. Den här parametern krävs på iOS. På Android måste parametern bara anges när mer än en Android-enhet eller -emulator är ansluten och körs på datorn. Om den specifierade enheten inte är ansluten, returnerar ADT avslutningskod 14: Enhetsfel (Andrid) eller Ogiltig enhet angiven (iOS). Om fler än en enhet eller emulator är ansluten, och ingen enhet anges, returnerar ADT avslutningskod 2: Användningsfel.

Obs! *Det går i AIR 3.4 och senare att installera en IPA-fil direkt i en iOS-enhet. För det krävs iTunes 10.5.0 eller senare.*

Använd kommandot `adt -devices` (finns i AIR 3.4 och senare) för att avgöra referens (handle) eller serienummer för anslutna enheter. Observera att på iOS ska du använda referensen (handle), inte enhetens UUID. Mer information finns i ”ADT-kommandot devices” på sidan 175.

På Android använder du Android ADB-verktyget för att lista serienumren på anslutna enheter och aktiva emulatorer:

```
adb devices
```

-package Filnamnet på det paket som ska installeras. På iOS måste det här vara en IPA-fil På Android måste detta vara ett APK-paket. Om det angivna paketet redan är installerat returnerar ADT felkod 14: Enhetsfel.

ADT-kommandot appVersion

Kommandot -appVersion rapporterar om den installerade versionen för ett program på en enhet eller emulator. Kommandot har följande syntax:

```
adt -appVersion -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform Namnet på enhetens plattform. Ange *ios* eller *android*.

-platformsdk Sökvägen till plattformens SDK för målenheten:

- Android – AIR 2.6+ SDK innehåller de verktyg från Android SDK som behövs för att implementera relevanta ADT-kommandon. Ange bara det här värdet om du vill använda en annan version av Android SDK. Om systemvariabeln AIR_ANDROID_SDK_HOME redan har angetts behöver sökvägen till plattforms-SDK inte anges på kommandoraden. (Om båda anges används den sökväg som anges på kommandoraden.)
- iOS – AIR SDK levereras med ett låst iOS SDK. Alternativet -platformsdk använder du för att förpacka program med en extern SDK så att du inte blir begränsad till att använda det iOS SDK som är låst. Om du till exempel har skapat ett tillägg med den senaste iOS SDK:n kan du ange detta SDK när du paketerar programmet. Dessutom när du använder ADT med iOS-simulatorn måste du alltid inkludera alternativet -platformsdk för att ange sökvägen till SDK:n för iOS-simulatorn.

-device Ange *ios_simulator* eller serienumret på enheten. Enheten behöver bara anges om fler än en Android-enhet eller emulator är ansluten till datorn och är igång. Om den angivna enheten inte är ansluten returnerar ADT avslutningskod 14: Enhetsfel. Om fler än en enhet eller emulator är ansluten, och ingen enhet anges, returnerar ADT avslutningskod 2: Användningsfel.

På Android använder du Android ADB-verktyget för att lista serienumren på anslutna enheter och aktiva emulatorer:

```
adb devices
```

-appid Det installerade programmets AIR-program-ID. Om inget program med det angivna ID:t är installerat på enheten returnerar ADT avslutningskod 14: Enhetsfel.

ADT-kommandot launchApp

Kommandot -launchApp kör ett installerat program på en enhet eller emulator. Kommandot har följande syntax:

```
adt -launchApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform Namnet på enhetens plattform. Ange *ios* eller *android*.

-platformsdk Sökvägen till plattformens SDK för målenheten:

- Android – AIR 2.6+ SDK innehåller de verktyg från Android SDK som behövs för att implementera relevanta ADT-kommandon. Ange bara det här värdet om du vill använda en annan version av Android SDK. Om systemvariabeln AIR_ANDROID_SDK_HOME redan har angetts behöver sökvägen till plattforms-SDK inte anges på kommandoraden. (Om båda anges används den sökväg som anges på kommandoraden.)
- iOS – AIR SDK levereras med ett låst iOS SDK. Alternativet -platformsdk använder du för att förpacka program med en extern SDK så att du inte blir begränsad till att använda det iOS SDK som är låst. Om du till exempel har skapat ett tillägg med den senaste iOS SDK:n kan du ange detta SDK när du paketerar programmet. Dessutom när du använder ADT med iOS-simulatorn måste du alltid inkludera alternativet -platformsdk för att ange sökvägen till SDK:n för iOS-simulatorn.

-device Ange *ios_simulator* eller serienumret på enheten. Enheten behöver bara anges om fler än en Android-enhet eller emulator är ansluten till datorn och är igång. Om den angivna enheten inte är ansluten returnerar ADT avslutningskod 14: Enhetsfel. Om fler än en enhet eller emulator är ansluten, och ingen enhet anges, returnerar ADT avslutningskod 2: Användningsfel.

På Android använder du Android ADB-verktyget för att lista serienumren på anslutna enheter och aktiva emulatorer:

```
adb devices
```

-appid Det installerade programmets AIR-program-ID. Om inget program med det angivna ID:t är installerat på enheten returnerar ADT avslutningskod 14: Enhetsfel.

ADT-kommandot uninstallApp

Kommandot -uninstallApp tar bort ett installerat program helt från en fjärrenhet eller emulator. Kommandot har följande syntax:

```
adt -uninstallApp -platform platformName -platformsdk path_to_sdk -device deviceID -appid
applicationID
```

-platform Namnet på enhetens plattform. Ange *ios* eller *android*.

-platformsdk Sökvägen till plattformens SDK för målenheten:

- Android – AIR 2.6+ SDK innehåller de verktyg från Android SDK som behövs för att implementera relevanta ADT-kommandon. Ange bara det här värdet om du vill använda en annan version av Android SDK. Om systemvariabeln AIR_ANDROID_SDK_HOME redan har angetts behöver sökvägen till plattforms-SDK inte anges på kommandoraden. (Om båda anges används den sökväg som anges på kommandoraden.)
- iOS – AIR SDK levereras med ett låst iOS SDK. Alternativet -platformsdk använder du för att förpacka program med en extern SDK så att du inte blir begränsad till att använda det iOS SDK som är låst. Om du till exempel har skapat ett tillägg med den senaste iOS SDK:n kan du ange detta SDK när du paketerar programmet. Dessutom när du använder ADT med iOS-simulatorn måste du alltid inkludera alternativet -platformsdk för att ange sökvägen till SDK:n för iOS-simulatorn.

-device Ange *ios_simulator* eller serienumret på enheten. Enheten behöver bara anges om fler än en Android-enhet eller emulator är ansluten till datorn och är igång. Om den angivna enheten inte är ansluten returnerar ADT avslutningskod 14: Enhetsfel. Om fler än en enhet eller emulator är ansluten, och ingen enhet anges, returnerar ADT avslutningskod 2: Användningsfel.

På Android använder du Android ADB-verktyget för att lista serienumren på anslutna enheter och aktiva emulatorer:

```
adb devices
```

-appid Det installerade programmets AIR-program-ID. Om inget program med det angivna ID:t är installerat på enheten returnerar ADT avslutningskod 14: Enhetsfel.

ADT-kommandot installRuntime

Kommandot -installRuntime installerar AIR-miljön på en enhet.

Du måste avinstallera den befintliga versionen av AIR-miljön innan du kan installera om med det här kommandot.

Kommandot har följande syntax:

```
adt -installRuntime -platform platformName -platformsdk path_to_sdk -device deviceID -package
fileName
```

-platform Namnet på enhetens plattform. Det här kommandot stöds för tillfället bara på Android-plattformen. Använd namnet *android*.

-platformsdk Sökvägen till plattformens SDK för målenheten. För tillfället är den enda plattforms-SDK som stöds Android. AIR 2.6+ SDK innehåller de verktyg från Android SDK som behövs för att implementera relevanta ADT-kommandon. Ange bara det här värdet om du vill använda en annan version av Android SDK. Om systemvariabeln AIR_ANDROID_SDK_HOME redan har angetts behöver sökvägen till plattforms-SDK inte anges på kommandoraden. (Om båda anges används den sökväg som anges på kommandoraden.)

-device Enhetens serienummer. Enheten behöver bara anges om fler än en enhet eller emulator är ansluten till datorn och igång. Om den angivna enheten inte är ansluten returnerar ADT avslutningskod 14: Enhetsfel. Om fler än en enhet eller emulator är ansluten, och ingen enhet anges, returnerar ADT avslutningskod 2: Användningsfel.

På Android använder du Android ADB-verktyget för att lista serienumren på anslutna enheter och aktiva emulatorer:

```
adb devices
```

-package Filnamnet på den miljö som ska installeras. På Android måste detta vara ett APK-paket. Om inget paket anges väljs lämplig miljö för enheten eller emulatorn från de som är tillgängliga i AIR SDK. Om miljön redan är installerad returnerar ADT felkod 14: Enhetsfel.

ADT-kommandot runtimeVersion

Kommandot -runtimeVersion rapporterar om den installerade versionen av AIR-miljön på en enhet eller emulator. Kommandot har följande syntax:

```
adt -runtimeVersion -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform Namnet på enhetens plattform. Det här kommandot stöds för tillfället bara på Android-plattformen. Använd namnet *android*.

-platformsdk Sökvägen till plattformens SDK för målenheten. För tillfället är den enda plattforms-SDK som stöds Android. AIR 2.6+ SDK innehåller de verktyg från Android SDK som behövs för att implementera relevanta ADT-kommandon. Ange bara det här värdet om du vill använda en annan version av Android SDK. Om systemvariabeln AIR_ANDROID_SDK_HOME redan har angetts behöver sökvägen till plattforms-SDK inte anges på kommandoraden. (Om båda anges används den sökväg som anges på kommandoraden.)

-device Enhetens serienummer. Enheten behöver bara anges om fler än en enhet eller emulator är ansluten till datorn och igång. Om miljön inte har installerats, eller om den angivna enheten inte är ansluten, returnerar ADT avslutningskod 14: Enhetsfel. Om fler än en enhet eller emulator är ansluten, och ingen enhet anges, returnerar ADT avslutningskod 2: Användningsfel.

På Android använder du Android ADB-verktyget för att lista serienumren på anslutna enheter och aktiva emulatorer:

```
adb devices
```

ADT-kommandot uninstallRuntime

Kommandot -uninstallRuntime tar bort AIR-miljön helt från en enhet eller emulator. Kommandot har följande syntax:

```
adt -uninstallRuntime -platform platformName -platformsdk path_to_sdk -device deviceID
```

-platform Namnet på enhetens plattform. Det här kommandot stöds för tillfället bara på Android-plattformen. Använd namnet *android*.

-platformsdk Sökvägen till plattformens SDK för målenheten. För tillfället är den enda plattforms-SDK som stöds Android. AIR 2.6+ SDK innehåller de verktyg från Android SDK som behövs för att implementera relevanta ADT-kommandon. Ange bara det här värdet om du vill använda en annan version av Android SDK. Om systemvariabeln AIR_ANDROID_SDK_HOME redan har angetts behöver sökvägen till plattforms-SDK inte anges på kommandoraden. (Om båda anges används den sökväg som anges på kommandoraden.)

-device Enhetens serienummer. Enheten behöver bara anges om fler än en enhet eller emulator är ansluten till datorn och igång. Om den angivna enheten inte är ansluten returnerar ADT avslutningskod 14: Enhetsfel. Om fler än en enhet eller emulator är ansluten, och ingen enhet anges, returnerar ADT avslutningskod 2: Användningsfel.

På Android använder du Android ADB-verktyget för att lista serienumren på anslutna enheter och aktiva emulatorer:

```
adb devices
```

ADT-kommandot devices

ADT-kommandot -help visar enhets-ID för mobilenheter och emulatorer som för tillfället är anslutna:

```
adt -devices -platform iOS|android
```

-platform Namnet på den plattform som ska kontrolleras. Ange `android` eller `iOS`.

Obs! *I iOS krävs detta kommando för iTunes 10.5.0 eller senare.*

ADT-kommandot help

ADT-kommandot -help visar kortfattad översikt över kommandoradsalternativen:

```
adt -help
```

Följande symbolkonventioner används i hjälpinformationen:

- <> – Objekt inom vinkelparenteser är uppgifter som du måste ange.
- () – Objekt inom parenteser är alternativ som behandlas som en grupp i utdata från hjälpkommandot.
- ALL_CAPS – Objekt som skrivs med versaler är en uppsättning alternativ som beskrivs separat.
- | – OR. Till exempel betyder (A | B) objekt A eller objekt B.
- ? – 0 eller 1. Ett frågetecken efter ett objekt anger att objektet är valfritt och att bara en instans kan förekomma, om det används.
- * – 0 eller fler. En asterisk efter ett objekt anger att objektet är valfritt och att vilket antal instanser som helst kan förekomma.
- + – 1 eller fler. Ett plustecken efter ett objekt anger att objektet är obligatoriskt och att flera instanser kan förekomma.
- ingen symbol – Om ett objekt inte har någon suffixsymbol är objektet obligatoriskt och bara en instans kan förekomma.

Uppsättningar med alternativ i ADT

Flera av ADT-kommandona delar gemensamma uppsättningar med alternativ.

ADT-kodsigneringsalternativ

ADT använder Java Cryptography Architecture (JCA) för att få tillgång till privata nycklar och certifikat för signering av AIR-program. Signeringsalternativen identifierar nyckelbehållaren och den privata nyckeln och certifikatet som finns i nyckelbehållaren.

Nyckelbehållaren måste inkludera både den privata nyckeln och den associerade certifikatkedjan. Om signeringscertifikatkedjan leder till ett betrott certifikat på en dator visas innehållet i certifikatets namnfält som utgivarnamn i dialogrutan för AIR-installation.

ADT kräver att certifikatet följer standarden x509v3 (RFC3280) och inkluderar tillägget för utökad nyckelanvändning med rätt värden för kodsignering. Begränsningar som definieras i certifikatet respekteras och kan utesluta användningen av vissa certifikat för signering av AIR-program.

Obs! *ADT använder proxyinställningarna för JRE (Java Runtime Environment), när det är lämpligt, för att ansluta till Internet-resurser och kontrollera listor över återkallade certifikat samt för att hämta tidstämplar. Om det uppstår problem när du ansluter till dessa Internet-resurser medan du använder ADT, och nätverket kräver specifika proxyinställningar, kanske du måste konfigurera JRE-proxyinställningarna.*

Syntax för signeringsalternativ i AIR

För signeringsalternativen används följande syntax (på en kommandorad):

```
-alias aliasName
-storetype type
-keystore path
-storepass password1
-keypass password2
-providerName className
-tsa url
```

-alias Aliaset för en nyckel i nyckelbehållaren. Det är inte nödvändigt att ange ett alias när en nyckelbehållare bara innehåller ett certifikat. Om inget alias anges används den första nyckeln i nyckelbehållaren.

Inte alla nyckelhanteringsprogram tillåter att ett alias tilldelas certifikat. När du till exempel använder Windows-systemnyckelbehållare, använder du certifikatets särskiljande namn som alias. Du kan använda verktyget Java Keytool om du vill visa vilka certifikat som är tillgängliga så att du kan bestämma alias. Om du till exempel kör kommandot:

```
keytool -list -storetype Windows-MY
```

skapas ett resultat ungefär som detta för ett certifikat:

```
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
```

Om du vill referera till det här certifikatet på ADT-kommandoraden ställer du in aliaset på:

```
CN=TestingCert,OU=QE,O=Adobe,C=US
```

I Mac OS X är aliaset för ett certifikat i Keychain det namn som visas i Nyckelhanteraren.

-storetype Typen av nyckelbehållare, som bestäms av nyckelbehållarimplementeringen. Standardimplementeringen av nyckelbehållaren som ingår i de flesta installationer av Java har stöd för typerna `JKS` och `PKCS12`. Java 5.0 har stöd för typen `PKCS11`, för åtkomst till nyckelbehållare i maskinvarutokens, och typen `Keychain` för åtkomst till nyckelkedjan i Mac OS X. Java 6.0 har stöd för typen `MSCAPI` (i Windows). Om andra JCA-providers har installerats och konfigurerats kan det finnas fler typer av nyckelbehållare. Om ingen nyckelbehållare anges används standardtypen för den förvalda JCA-providern.

Lagringstyp	Format på nyckelbehållare	Minimikrav på Java-version
JKS	Java-nyckelfil (.keystore)	1.2
PKCS12	Filen PKCS12 (.p12 eller .pfx)	1.4
PKCS11	Maskinvarutoken	1.5
KeychainStore	Mac OS X-nyckelkedja	1.5
Windows-MY eller Windows-ROOT	MSCAPI	1.6

-keystore Sökvägen till nyckelbehållarfilen för filbaserade lagringstyper.

-storepass Det lösenord som krävs för att få tillgång till nyckelbehållaren. Om du inte anger något lösenord uppmanas du att göra det.

-keypass Det lösenord som krävs för att få tillgång till den privata nyckel som används för att signera AIR-programmet. Om du inte anger något lösenord uppmanas du att göra det.

Obs! *Om du anger ett lösenord som en del av ADT-kommandot sparas lösenordstecknen i kommandoradshistoriken. Därför bör du inte använda alternativet -keypass eller -storepass när certifikatets säkerhet är viktig. Tänk även på att när du utesluter lösenordsalternativen visas inte de tecken du skriver när du anger lösenord (av samma säkerhetsskäl). Skriv lösenordet och tryck på Retur.*

-providerName JCA-providern för angiven typ av nyckelbehållare. Om detta inte anges används standardprovidern för den typen av nyckelbehållare.

-tsa Anger URL:en för en RFC3161-kompatibel tidstämpelserver som kan tidstämpla den digitala signaturen. Om ingen URL anges används den förvalda tidstämpelserver som tillhandhålls av Geotrust. När signaturen för ett AIR-program tidstämplas, kan programmet fortfarande installeras efter att signaturcertifikatet har upphört att gälla, eftersom tidstämpeln verifierar att certifikatet var giltigt vid signeringen.

Om ADT inte kan ansluta till tidstämpelservern avbryts signeringen och inget paket skapas. Ange `-tsa none` om du vill inaktivera tidstämpling. Ett AIR-programpaket utan tidstämpel kan emellertid inte längre installeras efter att certifikatet har upphört att gälla.

Obs! *Många av signeringsalternativen är likadana som motsvarande alternativ i Java-verktyget Keytool. Med Keytool kan du undersöka och hantera nyckelbehållare i Windows. I Mac OS X kan du också använda Apple®-säkerhetsverktyget för detta.*

-provisioning-profile Apple iOS-provisioneringsfilen. (Krävs endast för att paketera iOS-program.)

Exempel på signeringsalternativ

Signera med en P12-fil:

```
-storetype pkcs12 -keystore cert.p12
```

Signera med den förvalda Java-nyckelbehållare:

```
-alias AIRcert -storetype jks
```

Signera med en särskild Java-nyckelbehållaren:

```
-alias AIRcert -storetype jks -keystore certStore.keystore
```

Signera med Mac OS X-nyckelkedjan:

```
-alias AIRcert -storetype KeychainStore -providerName Apple
```

Signera med nyckelbehållaren i Windows-systemet:

```
-alias cn=AIRCert -storeype Windows-MY
```

Signera med en maskinvarutoken (information om hur du konfigurerar Java till att använda en token och rätt `providerName`-värde finns i tillverkarens instruktioner):

```
-alias AIRCert -storetype pkcs11 -providerName tokenProviderName
```

Signera utan att bädda in en tidstämpel:

```
-storetype pkcs12 -keystore cert.p12 -tsa none
```

Alternativ för filer och sökvägar

Alternativen för filer och sökvägar anger alla filer som ingår i paketet. Alternativen för filer och sökvägar har följande syntax:

```
files_and_dirs -C dir files_and_dirs -e file_or_dir dir -extdir dir
```

files_and_dirs De filer och kataloger som ska paketeras i AIR-filen. Du kan ange ett valfritt antal filer och kataloger, avgränsade med blanksteg. Om du gör en lista med kataloger, kommer alla filer och underkataloger däri, utom dolda filer, att läggas till i paketet. (Om programbeskrivningsfilen anges, antingen direkt eller via jokertecken eller katalogexpansion, ignoreras den och läggs inte till i paketet en andra gång.) Filer och kataloger som anges måste finnas i den aktuella katalogen eller i någon av underkatalogerna. Använd alternativet `-C` om du vill ändra den aktuella katalogen.

Viktigt! *Du kan inte använda jokertecken i `file_or_dir`-argumenten som finns efter alternativet `–C`. (Kommandoskal expanderar jokertecken innan argumenten skickas till ADT, vilket gör att ADT söker efter filer på fel plats.) Du kan emellertid fortfarande använda punkttecknet "." för aktuell katalog. Till exempel kopierar `-C assets .` allt i katalogen assets, inklusive eventuella underkataloger, till programpaketets rotnivå.*

`-C dir files_and_dirs` Ändrar arbetskatalogen till värdet på *dir* innan efterföljande filer och kataloger som lagts till i programpaketet (angivna i **`files_and_dirs`**) behandlas. Filerna eller katalogerna läggs till i programpaketets rot. Alternativet `–C` kan användas ett valfritt antal gånger för att inkludera filer från flera platser i filsystemet. Om du anger en relativ sökväg för `dir` löses sökvägen alltid från den ursprungliga arbetskatalogen.

Allt eftersom ADT bearbetar filer och kataloger som ingår i paketet, lagras de relativa sökvägarna mellan den aktuella katalogen och målfilerna. Dessa sökvägar expanderas till programkatalogstrukturen när paketet installeras. Om du anger `-C release/bin lib/feature.swf` placeras filen `release/bin/lib/feature.swf` i underkatalogen `lib` till rotprogrammappen.

`-e file_or_dir dir` Placerar filen eller katalogen i den angivna paketkatalogen. Detta alternativ kan inte användas när du paketerar en ANE-fil.

Obs! *Elementet `<content>` i programbeskrivningsfilen måste ange den slutgiltiga platsen för huvudprogramfilen i programpaketets katalogträd.*

-extdir *dir* Värdet för `dir` är namnet på en katalog där du ska söka efter systemspecifika tillägg (ANE-filer). Ange antingen en absolut eller en relativ sökväg till den aktuella katalogen. Du kan använda alternativet `-extdir` flera gånger.

Den angivna katalogen innehåller ANE-filer för systemspecifika tillägg som används i programmet. Varje ANE-fil i den här katalogen har filnamnstillägget .ane. Filnamnet före tillägget .ane behöver *inte* matcha värdet för elementet `extensionID` i programbeskrivningsfilen.

Om du till exempel använder `-extdir ./extensions`, kan katalogen `extensions` ha följande utseende:

```
extensions/
    extension1.ane
    extension2.ane
```

Obs! *Användningen av alternativet -extdir skiljer sig åt mellan ADT- och ADL-verktygen. I ADL anger alternativen en katalog som innehåller underkataloger, där var och en innehåller en ouppackad ANE-fil. I ADT anger alternativen en katalog som innehåller ANE-filer.*

Anslutningsalternativ för felsökning

När målet för paketet är apk-debug, ipa-debug eller ipa-debug-interpreter, kan anslutningsalternativen användas för att ange om programmet ska försöka ansluta till ett fjärrfelsökningsprogram (används ofta för wifi-felsökning) eller avlyssna en inkommande anslutning från ett fjärrfelsökningsprogram (används ofta för USB-felsökning). Använd alternativet `-connect` för att ansluta till en felsökare; använd alternativet `-listen` för att acceptera en anslutning från en felsökare via en USB-anslutning. Dessa alternativ utesluter varandra, dvs. du kan inte använda båda samtidigt.

Alternativet -connect har följande syntax:

```
-connect hostString
```

-connect Om det används försöker programmet ansluta till en fjärrfelsökare.

hostString En sträng som identifierar den dator som kör Flash-felsökningsverktyget FDB. Om det inte anges försöker programmet ansluta till en felsökare som körs på den dator där paketet skapas. hostString kan vara ett fullständigt kvalificerat datornamn med domän, som *datornamn.delgrupp.exempel.com*, eller en IP-adress, som *192.168.4.122*. Om det inte går att hitta den angivna datorn (eller standarddatorn) visas en dialogruta där ett giltigt värdnamn ska anges.

Alternativet `-listen` har följande syntax:

```
-listen port
```

-listen Om det finns väntar miljön på en anslutning från en fjärrfelsökare.

port (Valfritt) Den port som ska avlyssnas. Miljön avlyssnar som standard port 7936. Mer information om hur du använder alternativet `-listen` finns i ”Fjärrfelsökning med FDB via USB” på sidan 104.

Profileringsalternativ för Android-program

När paketets mål är apk-profile kan profileringsalternativen användas för att ange vilken förinläsar-SWF-fil som ska användas för prestanda och minnesprofilering. Profileringsalternativen har följande syntax:

```
-preloadSWFPath directory
```

-preloadSWFPath Om det används försöker programmet hitta förinläsar-SWF-filen i den angivna katalogen. Om det inte anges inkluderas förinläsar-SWF-filen från AIR SDK.

directory Den katalog som innehåller förinläsar-SWF-filen för profileringen.

Alternativ för ANE (Native Extension)

Med ANE-alternativen anges alternativ och filer för paketering av en ANE-fil för ett systemspecifikt tillägg. Använd dessa alternativ med ett ADT-paketkommando där `-target`-alternativet är `ane`.

```
extension-descriptor -swc swcPath
   -platform platformName
  -platformoptions path/platform.xml
   FILE_OPTIONS
```

extension-descriptor Beskrivningsfilen för ANE-tillägget.

-swc Den SWC-fil som innehåller ActionScript-koden och resurserna för ANE-tillägget.

-platform Namnet på den plattform som den här ANE-filen har stöd för. Du kan inkludera flera `-platform`-alternativ där vart och ett har egna `FILE_OPTIONS`.

-platformoptions Sökvägen till plattformsalternativ (filen platform.xml). Använd den här filen för att ange icke standardiserade länkningsalternativ, delade bibliotek och statiska tredjepartsbibliotek som används i tillägget. Mer information och exempel finns i Systemspecifika iOS-bibliotek.

FILE_OPTIONS Identifierar de systemspecifika plattformsfilerna som ska inkluderas i paketet, till exempel statiska bibliotek som ska inkluderas i ANE-tilläggspaketet. Filalternativen beskrivs mer ingående i ”Alternativ för filer och sökvägar” på sidan 178. (Observera att alternativet -e inte kan användas vid paketering av en ANE-fil.)

Fler hjälpavsnitt

Paketering av ett ANE-tillägg

ADT-felmeddelanden

I följande tabeller visas möjliga fel, och sannolika orsaker, som kan rapporteras av ADT-programmet:

Felområde: programbeskrivningsvalidering

Felkod nr.	Beskrivning	Anteckningar
100	Det går inte att tolka programbeskrivningen.	Kontrollera om programbeskrivningen innehåller XML-syntaxfel, t.ex. sluttaggar som saknas.
101	Ett namnutrymme saknas	Lägg till namnutrymmet som saknas.
102	Ogiltigt namnutrymme	Kontrollera att namnutrymmet är rätt stavat.
103	Oväntat element eller attribut	Ta bort felaktiga element och attribut. Anpassade värden tillåts inte i beskrivningsfilen. Kontrollera att element- och attributnamnen är rätt stavade. Kontrollera att elementen är placerade i rätt överordnat element och att attributen används med rätt element.
104	Ett element eller attribut saknas	Lägg till elementet eller attributet som saknas.

Felkod nr.	Beskrivning	Anteckningar
105	Ett element eller attribut innehåller ett ogiltigt värde	Korrigera det felaktiga värdet.
106	Ogiltig kombination av fönsterattribut	En del fönsterinställningar, t.ex. `transparency = true` och `systemChrome = standard`, kan inte användas tillsammans. Ändra en av de inkompatibla inställningarna.
107	Den minsta fönsterstorleken är större än den största tillåtna fönsterstorleken	Ändra inställningen för största eller minsta fönsterstorlek.
108	Attributet används redan i ett tidigare element	
109	Dubblettelement.	Ta bort dubblettelementet.
110	Minst ett element av angiven typ krävs.	Lägg till det element som saknas.
111	Ingen av de profiler som listas i programbeskrivningen har stöd för ANE-tillägg.	Lägg till en profil i listan supportedProfiles med stöd för ANE-tillägg (Native Extensions).
112	AIR-målet saknar stöd för ANE-tillägg.	Välj ett mål som har stöd för ANE-tillägg.
113	<nativeLibrary> och <initializer> måste användas tillsammans.	En initierarfunktion måste anges för varje systemspecifikt bibliotek i ANE-tillägget.
114	<finalizer> utan <nativeLibrary> hittades.	Ange inte finalizer om inte plattformen använder ett systemspecifikt bibliotek.
115	Standardplattformen behöver inte innehålla någon systemspecifik implementering.	Ange inte något systemspecifikt bibliotek i standardplattformselementet.
116	Webbläsaranrop stöds inte för detta mål.	Elementet `<allowBrowserInvocation>` får inte vara `true` för det angivna paketeringsmålet.
117	För det här målet krävs minst namnutrymmet n för att paketera ANE-tillägg.	Ändra AIR-namnutrymmet i programbeskrivningen till ett värde som stöds.

Mer information om namnutrymmen, element, attribut och deras giltiga värden finns i avsnittet ”AIR-programbeskrivningsfiler” på sidan 200.

Felområde: programikon

Felkod nr.	Beskrivning	Anteckningar
200	Det går inte att öppna ikonfilen	Kontrollera att filen finns på angiven sökväg. Kontrollera att filen kan öppnas genom att öppna den med ett annat program.
201	Ikonen har fel storlek	Ikonstorleken (i antal pixlar) måste matcha XML-taggen. Exempel (med angivet programbeskrivningselement): `<image32x32>icon.png</image32x32>` Bilden i `icon.png` måste vara exakt 32 × 32 pixlar.
202	Ikonfilen innehåller ett bildformat som inte stöds	Endast PNG-formatet stöds. Konvertera bilder i andra format innan du paketerar programmet.

Felområde: programfil

Felkod nr.	Beskrivning	Anteckningar
300	En fil saknas eller kan inte öppnas	Det går inte att hitta eller öppna en fil som angetts på kommandoraden.
301	Programbeskrivningsfilen saknas eller kan inte öppnas	Programbeskrivningsfilen kan varken öppnas eller hittas på den angivna sökvägen.
302	Filen med rotinnehåll saknas i paketet	SWF- eller HTML-filen som `<content>`-elementet i programbeskrivningen refererar till måste läggas till i paketet genom att inkluderas bland filerna som anges på ADT-kommandoraden.
303	Ikonfilen saknas i paketet	Ikonfilerna som anges i programbeskrivningen måste läggas till i paketet genom att inkluderas bland filerna som anges på ADT-kommandoraden. Ikonfilerna läggs inte till automatiskt.
304	Det ursprungliga fönsterinnehållet är ogiltigt	Filen som `<content>`-elementet i programbeskrivningen refererar till är inte en giltig HTML- eller SWF-fil.
305	SWF-versionen för det ursprungliga fönsterinnehållet stöds inte av namnutrymmesversionen	SWF-versionen för filen som `<content>`-elementet i programbeskrivningen refererar till stöds inte av den AIR-version som anges i beskrivningens namnutrymme. Det här felet genereras exempelvis om du försöker paketera en SWF10-fil (Flash Player 10) som det ursprungliga innehållet i ett AIR 1.1-program.
306	Profilen stöds inte.	Den profil som du angav i programbeskrivningsfilen stöds inte. Läs mer i ”supportedProfiles” på sidan 234.
307	Namnutrymmet måste vara minst *nnn*.	Använd lämpligt namnutrymme för de funktioner som används i programmet (t.ex. 2.0-namnutrymmet).

Avslutningskoder för andra fel

Avslutningskod	Beskrivning	Anteckningar
2	Användningsfel	Kontrollera kommandoradsargumenten för att se om det finns något fel.
5	Okänt fel	Det här felet indikerar en situation som inte har en självklar förklaring. Möjliga rotorsaker är inkompatibilitet mellan ADT och Java Runtime Environment, skadade ADT- eller JRE-installationer och programmeringsfel i ADT.
6	Det gick inte att skriva till utdatakatalogen	Kontrollera att den angivna (eller underförstådda) utdatakatalogen kan nås och att det finns tillräckligt med diskutrymme på disken som katalogen finns på.
7	Det gick inte att komma åt certifikatet	Kontrollera att sökvägen till nyckelbehållaren stämmer. Kontrollera att certifikatet i nyckelbehållaren kan nås. Du kan felsöka certifikatåtkomstproblem med verktyget Java 1.6 Keytool.
8	Ogiltigt certifikat	Certifikatfilen har fel format, har modifierats, gått ut eller återkallats.
9	Det gick inte att signera AIR-filen	Kontrollera signeringsalternativen som skickas till ADT.
10	Det gick inte att skapa tidsstämpeln	ADT kunde inte upprätta en anslutning till tidsstämpelservern. Om du ansluter till Internet via en proxyserver kan du behöva konfigurera JRE-proxyinställningarna.
11	Certifikatgenereringsfel	Kontrollera argumenten på kommandoraden som används för att skapa signaturer.
12	Ogiltiga indata	Kontrollera filsökvägar och andra argument som skickas till ADT på kommandoraden.
13	Enhets-SDK saknas	Kontrollera enhetens SDK-konfiguration. Det går inte att hitta enhets-SDK som krävs för att köra det angivna kommandot.
14	Enhetsfel	Det går inte att köra kommandot på grund av en enhetsbegränsning eller ett enhetsproblem. Den här koden skickas till exempel vid försök att avinstallera ett program som inte är installerat.

Avslutningskod	Beskrivning	Anteckningar
15	Ingen enhet	Kontrollera att det finns en enhet som är ansluten och igång eller att en emulator körs.
16	GPL-komponenter saknas	Den aktuella AIR SDK-versionen inkluderar inte alla komponenter som behövs för att utföra den begärda åtgärden.
17	Verktyget för enhetspaketering misslyckades.	Paketet kunde inte skapas eftersom en förväntad komponent från operativsystemet saknas.

Android-fel

Avslutningskod	Beskrivning	Anteckningar
400	Den aktuella Android SDK-versionen saknar stöd för attribut.	Kontrollera att attributnamnet är rättstavat och giltigt för det element i vilket det finns. Du kan behöva ange flaggan -platformsdk i ADT-kommandot om attributet har introducerats efter Android 2.2.
401	Den aktuella Android SDK-versionen saknar stöd för attributvärde.	Kontrollera att attributvärdet är rättstavat och giltigt för det attributet. Du kan behöva ange flaggan -platformsdk i ADT-kommandot om attributvärdet har introducerats efter Android 2.2.
402	Den aktuella Android SDK-versionen saknar stöd för XML-taggen.	Kontrollera att XML-taggens namn är rättstavat och giltigt som dokumentelement i Androids manifestfil. Du kan behöva ange flaggan -platformsdk i ADT-kommandot om elementet har introducerats efter Android 2.2.
403	Android-taggen kan inte åsidosättas	Programmet försöker åsidosätta ett element i Androids manifestfil som reserverats för AIR. Läs mer i "Android-inställningar" på sidan 73.
404	Android-attribut kan inte åsidosättas	Programmet försöker åsidosätta ett attribut i Androids manifestfil som reserverats för AIR. Läs mer i "Android-inställningar" på sidan 73.
405	Android-taggen %1 måste vara det första elementet i taggen manifestAdditions	Flytta den angivna taggen till den obligatoriska platsen.
406	Attributet %1 i Android-taggen %2 har det ogiltiga värdet %3.	Ange ett giltigt värde för attributet.

ADT-systemvariabler

ADT läser värdena för följande systemvariabler (om de har angetts):

AIR_ANDROID_SDK_HOME anger sökvägen till rotkatalogen för Android SDK (den katalog som innehåller mappen tools). AIR 2.6+ SDK innehåller de verktyg från Android SDK som behövs för att implementera relevanta ADT-kommandon. Ange bara det här värdet om du vill använda en annan version av Android SDK. Om den här variabeln anges behöver alternativet -platformsdk inte anges även om ADT-kommandon som behöver det körs. Om både den här variabeln och kommandoradsalternativet anges används den sökväg som anges på kommandoraden.

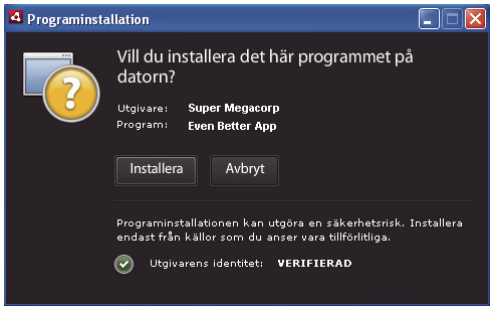
AIR_EXTENSION_PATH anger en kataloglista för att söka efter de ANE-tillägg som krävs för ett program. Kataloglistan söks igenom för att hitta ANE-tilläggskataloger som anges på kommandoraden i ADT. I ADL-kommandot används även denna miljövariabel.

Obs! *På vissa system kan dubbelbytetecken i filsystemsökvägar feltolkas när de lagras i dessa systemvariabler. Om detta händer provar du med att ange teckenuppsättningen UTF-8 för den JRE-version som används för att köra ADT. Detta utförs som standard i det skript som används för att starta ADT på Mac och Linux. I Windows-filen adt.bat, eller om du kör ADT direkt från Java, anger du alternativet* `-Dfile.encoding=UTF-8` *på Java-kommandoraden.*

Kapitel 13: Signera AIR-program

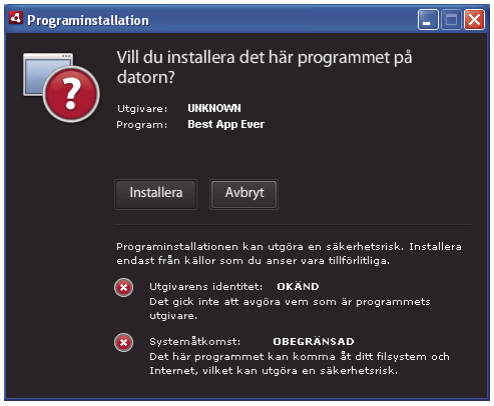
Signera en AIR-fil digitalt

Om du signerar dina AIR-installationsfiler digitalt med ett certifikat som har utfärdats av en erkänd certifikatutfärdare, kan du garantera dina användare att programmet de installerar inte har ändrats av misstag eller på ett skadligt sätt, och det identifierar dig som signerare (utgivare). AIR visar utgivarens namn under installationen när AIR-programmet har signerats med ett certifikat som är tillförlitligt, eller som skapar en *kedja* till ett certifikat som är betrott på installationsdatorn:



Bekräftande installationsdialogruta för program som signeras av ett betrott certifikat

Om du signerar ett program med ett självsignerande certifikat (eller ett certifikat som inte kedjas till ett betrott certifikat) måste användaren ta en större risk när programmet installeras. Installationsdialogrutorna återspeglar den högre risken:



Bekräftande installationsdialogruta för program som signeras av ett självsignerat certifikat

***Viktigt!** En oärlig person kan förfalska en AIR-fil med din identitet om han eller hon på något sätt kan hämta din signeringsnyckelfil eller upptäcker din privata nyckel.*

Kodsigneringscertifikat

Säkerhetsgarantier, begränsningar och juridiska åtaganden när det gäller användning av kodsigneringscertifikat beskrivs i sekretesspolicyn och abonnemangsavtalen som publiceras av certifikatutfärdaren. Mer information om avtalen för certifikatutfärdare som utfärdar signeringscertifikat för AIR-kod finns på:

ChosenSecurity (http://www.chosensecurity.com/products/tc_publisher_id_adobe_air.htm)

ChosenSecurity CPS (http://www.chosensecurity.com/resource_center/repository.htm)

GlobalSign (http://www.globalsign.com/code-signing/index.html)

GlobalSign CPS (http://www.globalsign.com/repository/index.htm)

Thawte CPS (http://www.thawte.com/cps/index.html)

VeriSign CPS (http://www.verisign.com/repository/CPS/)

VeriSign Subscriber's Agreement (https://www.verisign.com/repository/subscriber/SUBAGR.html)

Om kodsignering i AIR

När en AIR-fil har signerats inkluderas en digital signatur i installationsfilen. Signaturen omfattar en sammanfattning av paketet, som används för att verifiera att AIR-filen inte har ändrats sedan den signerades, och den innehåller information om signeringscertifikatet, som används för att verifiera utgivarens identitet.

AIR använder den offentliga nyckelstrukturen som stöds i operativsystemets certifikatarkiv för att ta reda på om ett certifikat är betrott eller inte. Datorn där AIR-programmet är installerat måste antingen lita direkt på certifikatet som används för att signera AIR-programmet, eller lita på en kedja av certifikat som länkar certifikatet till en betrodd certifikatutfärdare för att informationen om utgivaren ska kunna verifieras.

Om en AIR-fil signeras med ett certifikat som inte länkar till något betrott rotcertifikat (vanligtvis omfattar detta alla självsignerade certifikat), kan informationen om utgivaren inte verifieras. AIR kan ta reda på om ett AIR-paket inte har ändrats sedan det signerades, men det går inte att ta reda på vem som verkligen skapade och signerade filen.

Obs! *En användare kan välja att lita på ett självsignerat certifikat. Därefter visar alla AIR-program som har signerats med certifikatet värdet i fältet för vanligt namn som utgivarens namn. En användare kan inte ange att ett certifikat är betrott. Certifikatet (utan den privata nyckeln) måste anges för användaren separat och användaren måste använda någon av metoderna i operativsystemet eller ett lämpligt verktyg för att importera certifikatet till rätt plats i systemcertifikatarkivet.*

Om utgivaridentifierare i AIR

Viktigt! *Från och med AIR 1.5.3 används inte utgivar-ID längre och beräknas inte längre utifrån kodsigneringscertifikatet. Nya program behöver inte och bör inte använda ett utgivar-ID. När du uppdaterar befintliga program måste du ange ursprungligt utgivar-ID i programbeskrivningsfilen.*

Före AIR 1.5.3 genererade installationsprogrammet för AIR-programmet ett utgivar-ID under installationen av en AIR-fil. Detta var ett ID som är unikt för certifikatet som används för att signera AIR-filen. Om du återanvände samma certifikat för flera AIR-program fick de samma utgivar-ID. Om du signerade en programuppdatering med ett annat certifikat och ibland till och med en förnyad instans av det ursprungliga certifikatet ändrades utgivar-ID.

I AIR 1.5.3 och senare tilldelas inte ett utgivar-ID av AIR. Ett program som publiceras med AIR 1.5.3 kan ange en sträng för utgivar-ID i programbeskrivningen. Du bör endast ange ett utgivar-ID när du publicerar uppdateringar till program som ursprungligen har publicerats för tidigare versioner av AIR än 1.5.3. Om du inte anger ursprungligt ID i programbeskrivningen hanteras inte det nya AIR-paketet som en uppdatering av det befintliga programmet.

Du hittar ursprungligt utgivar-ID i filen `publisherid` i undermappen META-INF/AIR där det ursprungliga programmet är installerat. Strängen i den filen är utgivar-ID:t. Programbeskrivningen måste ange runtime-versionen av AIR 1.5.3 (eller senare) i namnutrymmesdeklarationen i programbeskrivningsfilen för att ange utgivar-ID manuellt.

Eventuellt utgivar-ID används i följande syften:

- Som en del av krypteringsnyckeln för den krypterade lokala lagringsplatsen
- Som en del av sökvägen till programlagringskatalogen
- Som en del av anslutningssträngen för lokala anslutningar
- Som en del av identitetssträngen som används för att anropa ett program med webbläsar-API:n för AIR
- Som en del av OSID (används när man skapar anpassade installations-/avinstallationsprogram)

När ett utgivar-ID ändras, ändras även beteendet för eventuella AIR-funktioner som är beroende av ID:t. Det går till exempel inte längre att komma åt data på den befintliga krypterade lokala lagringsplatsen och eventuella Flash- eller AIR-instanser som skapar en lokal anslutning till programmet måste använda det nya ID:t i anslutningssträngen. Det går inte att ändra utgivar-ID:t för ett installerat program i AIR 1.5.3 eller senare. Om du använder ett annat utgivar-ID när du publicerar ett AIR-paket hanterar installationsprogrammet det nya paketet som ett annat program i stället för som en uppdatering.

Om certifikatformat

AIR-signeringsverktygen accepterar alla nyckelbehållare som är tillgängliga via Java Cryptography Architecture (JCA). Detta omfattar nyckelbaserade nyckelbehållare som filer i PKCS12-format (som vanligtvis använder filnamnstillägget .pfx eller .p12), Java .keystore-filer, PKCS11-maskinvarunycklar och systemnycklar. Vilka nyckelformat som ADT kan använda är beroende av den version och konfiguration av Java Runtime som används för att köra ADT. För att kunna använda vissa typer av nyckelbehållare, till exempel PKCS11-maskinvarutokens, måste du kanske installera och konfigurera andra programdrivrutiner och JCA-plug-iner.

För signering av AIR-filer kan du använda de flesta befintliga kodsigneringscertifikat eller skaffa ett nytt som utfärdats för signering av AIR-program. Till exempel kan du använda någon av följande typer av certifikat från VeriSign, Thawte, GlobalSign eller ChosenSecurity:

- ChosenSecurity
  - TC Publisher ID för Adobe AIR
- GlobalSign
  - ObjectSign Code Signing Certificate
- Thawte:
  - AIR Developer Certificate
  - Apple Developer Certificate
  - JavaSoft Developer Certificate
  - Microsoft Authenticode Certificate
- VeriSign:
  - Adobe AIR Digital ID
  - Microsoft Authenticode Digital ID
  - Sun Java Signing Digital ID

Obs! *Certifikatet måste vara skapat för kodsignering. Du kan inte använda SSL eller andra typer av certifikat för att signera AIR-filer.*

Tidstämplar

När du signerar en AIR-fil frågar paketeringsverktyget servern om det finns en tidstämpelutfärdare för att kunna hämta oberoende verifierbart datum och tid för signering. Tidstämpeln som hämtas bäddas in i AIR-filen. Om signeringscertifikatet är giltigt vid tiden för signering, kan AIR-filen installeras även efter att certifikatet har upphört att gälla. Om ingen tidstämpel hämtas, kan AIR-filen å andra sidan inte installeras när certifikatet har upphört att gälla eller återkallats.

AIR-paketeringsverktyget hämtar som standard en tidstämpel. Om du vill att programmen ska kunna paketeras även när tidstämpeltjänsten inte är tillgänglig, kan du stänga av tidstämpling. Adobe rekommenderar att alla AIR-filer som distribueras offentligt innehåller en tidstämpel.

Den standardutfärdare för tidstämpling som används av AIR-paketeringsverktyget är Geotrust.

Hämta ett certifikat

Om du vill hämta ett certifikat ska du vanligtvis besöka certifikatutfärdarens webbplats och gå igenom företagets anskaffningsprocess. Vilka verktyg som används för att skapa nyckelfilen som behövs av AIR-verktygen beror på vilken typ av certifikat som du köper, hur certifikatet lagras på den mottagande datorn och, i vissa fall, vilken webbläsare som används för att hämta certifikatet. Om du till exempel vill skaffa och exportera ett Adobe Developer-certifikat från Thawte måste du använda Mozilla Firefox. Certifikatet kan sedan exporteras som en P12- eller PFX-fil direkt från användargränssnittet i Firefox.

Obs! *I Java-versionerna 1.5 och senare går det inte att använda hög-ASCII-tecken i lösenord för att skydda PKCS12-certifikatfiler. Java används i AIR-utvecklingsverktyg för att skapa signerade AIR-paket. När du exporterar certifikatet som en .p12- eller .pfx-fil ska du endast använda reguljära ASCII-tecken i lösenordet.*

Du kan skapa ett självsignerat certifikat med hjälp av Air-utvecklingsverktyget som används för att paketera AIR-installationsfilerna. Du kan också använda en del verktyg från andra företag.

Instruktioner om hur du skapar ett självsignerat certifikat, samt hur du signerar en AIR-fil, finns i ”AIR Developer Tool (ADT)” på sidan 162. Du kan också exportera och signera AIR-filer med Flash Builder, Dreamweaver och AIR-uppdateringen för Flash.

I exemplen nedan beskrivs hur du hämtar ett AIR-utvecklarcertifikat från certifikatutfärdaren Thawte och förbereder det för ADT.

Exempel: Hämta ett AIR-utvecklarcertifikat från Thawte

Obs! *I det här exemplet illustreras bara ett av de många sätt som du kan hämta och förbereda ett kodsigneringscertifikat på. Varje certifikatutfärdare har egna policyer och processer.*

Om du vill köpa ett AIR-utvecklarcertifikat på webbplatsen för Thawte måste du använda webbläsaren Mozilla Firefox. Den privata nyckeln för certifikaten lagras i webbläsarens nyckelbehållare. Se till att Firefox-nyckelbehållaren skyddas med ett huvudlösenord och att själva datorn är fysiskt säker. (Du kan exportera och ta bort certifikatet och den privata nyckeln från webbläsarens nyckelbehållare när anskaffningsprocessen är färdig.)

När du skaffar certifikatet skapas ett privat/offentligt nyckelpar. Den privata nyckeln lagras automatiskt i nyckelbehållaren i Firefox. Du måste använda samma dator och webbläsare för att begära och hämta certifikatet från Thawtes webbplats.

1 Besök Thawte-webbplatsen och navigera till produktsidan för kodsigneringscertifikat.

2 Välj Adobe AIR Developer Certificate i listan med kodsigneringscertifikat.

3 Gå igenom anmälningsprocessen i tre steg. Du måste ange organisations- och kontaktinformation. Thawte kontrollerar sedan identiteten och kan begära mer information. När kontrollen är klar, skickar Thawte ett e-postmeddelande med instruktioner för hur du hämtar certifikatet.

Obs! *Mer information om vilken typ av dokument som kan behövas finns här: https://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/enroll_codesign_eng.pdf.*

4 Hämta det utfärdade certifikatet från Thawte-webbplatsen. Certifikatet sparas automatiskt i Firefox-nyckelbehållaren.

5 Exportera en nyckelfil som innehåller den privata nyckeln och certifikatet från Firefox-nyckelbehållaren så här:

Obs! *När du exporterar den privata nyckeln/certifikatet från Firefox exporteras den i ett P12-format (pfx) som ADT, Flex, Flash och Dreamweaver kan använda.*

a Öppna Firefox-dialogrutan *Certificate Manager*:

b I Windows: öppna Verktyg -> Alternativ -> Avancerat -> Kryptering -> Visa certifikat

c I Mac OS: öppna Firefox -> Inställningar -> Avancerat -> Kryptering -> Visa certifikat

d I Linux: öppna Redigera -> Inställningar -> Avancerat -> Kryptering -> Visa certifikat

e Markera Adobe AIR Code Signing Certificate i listan med certifikat och klicka på **Backup**.

f Skriv ett filnamn och den plats som nyckelfilen ska exporteras till och klicka på **Save**.

g Om du använder Firefox-huvudlösenordet ombeds du att skriva lösenordet för programsäkerhetsenheten så att du kan exportera filen. (Det här lösenordet används bara i Firefox.)

h Skapa ett lösenord för nyckelfilen i dialogrutan *Choose a Certificate Backup Password*.

Viktigt! *Det här lösenordet skyddar nyckelfilen och krävs när filen används för signering av AIR-program. Välj ett säkert lösenord.*

i Klicka på OK. Ett meddelande om att lösenordet har säkerhetskopierats bör visas. Nyckelfilen som innehåller den privata nyckeln och certifikatet sparas med filnamnstillägget .p12 (i PKCS12-format)

6 Använd den exporterade nyckelfilen med ADT, Flash Builder, Flash Professional eller Dreamweaver. Lösenordet som skapas för filen krävs när ett AIR-program signeras.

Viktigt! *Den privata nyckeln och certifikatet finns fortfarande kvar i nyckelbehållaren i Firefox. Då kan du exportera en extra kopia av certifikatfilen, men det innebär också ytterligare ett sätt att få tillgång till filen som måste skyddas så att certifikatet och den privata nyckeln också skyddas.*

Byta certifikat

Under vissa omständigheter måste du byta det certifikat du använder för att signera uppdateringar till ditt AIR-program. Detta kan gälla när du vill:

- Förnya det ursprungliga signeringscertifikatet.
- Uppgradera från ett självsignerat certifikat till ett certifikat som utfärdas av en certifikatutfärdare
- Byta från ett självsignerat certifikat som snart upphör att gälla
- Byta från ett kommersiellt certifikat till ett annat, till exempel när företagets identitet ändras

För att AIR ska identifiera en AIR-fil som en uppdatering måste du antingen signera både ursprungliga och uppdaterade AIR-filer med samma certifikat eller tillämpa en certifikatflyttningssignatur på uppdateringen. En flyttningssignatur är en andra signatur som tillämpas på det uppdaterade AIR-paketet med det ursprungliga certifikatet. Flyttningssignaturen använder det ursprungliga certifikatet för att ange att signeraren är programmets ursprungliga utgivare.

När en AIR-fil med en flyttningssignatur har installerats blir det nya certifikatet primärt certifikat. Efterföljande uppdateringar kräver ingen flyttningssignatur. Du bör emellertid använda flyttningssignaturer så länge som möjligt för att tillgodose användare som hoppar över uppdateringar.

Viktigt! *Du måste byta certifikat och använda en flyttningssignatur på uppdateringen med originalcertifikatet innan det upphör att gälla. Annars måste användarna avinstallera den befintliga versionen av ditt program innan de kan installera den nya versionen. För AIR 1.5.3 och senare kan du använda en flyttningssignatur med ett certifikat som har upphört att gälla under en 365-dagars frist efter att det gått ut. Du kan däremot inte använda det utgångna certifikatet för att använda den huvudsakliga programsignaturen.*

Så här byter du certifikat:

1. Skapa en uppdatering av programmet.
2. Paketera och signera AIR-uppdateringsfilen med det **nya** certifikatet.
3. Signera AIR-filen igen med det **ursprungliga** certifikatet (med hjälp av ADT-kommandot `-migrate`).

En AIR-fil med en flyttningssignatur är i övrigt som en normal AIR-fil. Om programmet installeras i ett system utan den ursprungliga versionen, installeras den nya versionen på vanligt sätt.

Obs! *Före AIR 1.5.3 krävdes inte alltid en flyttningssignatur för att signera ett AIR-program med ett förnyat certifikat. Från och med AIR 1.5.3 krävs alltid en flyttningssignatur för förnyade certifikat.*

Om du vill använda en flyttningssignatur använder du ”ADT-kommandot migrate” på sidan 170, enligt beskrivningen i ”Signera en uppdaterad version av ett AIR-program” på sidan 195.

Obs! *ADT-kommandot migrate kan inte användas med AIR-program för datorer som inkluderar systemspecifika tillägg, eftersom sådana program paketeras som systemspecifika installationsprogram och inte som .air-filer. Om du vill ändra certifikat för ett AIR-program för datorer som inkluderar ett systemspecifikt tillägg paketerar du programmet med ”ADT-kommandot package” på sidan 163 och flaggan -migrate.*

Ändringar av programidentitet

Före AIR 1.5.3 ändrades ett AIR-programs identitet när en uppdatering signerad med en flyttningssignatur installerades. En ändring av ett programs identitet har flera följder, inklusive:

- Den nya programversionen kan inte få tillgång till data på den befintliga krypterade lokala lagringsplatsen.
- Platsen för programlagringskatalogen ändras. Data på den gamla platsen kopieras inte till den nya katalogen. (Men det nya programmet kan hitta den ursprungliga katalogen utifrån det gamla utgivar-ID:t.)
- Programmet kan inte längre öppna lokala anslutningar med det tidigare utgivar-ID:t.
- Identitetssträngen som används för att komma åt ett program från en webbsida ändras.
- Programmets OSID ändras. (OSID används när man skriver anpassade installations-/avinstallationsprogram.)

När en uppdatering publiceras med AIR 1.5.3 eller senare versioner kan inte programmets identitet ändras. Ursprungligt program- och utgivar-ID måste anges i den uppdaterade AIR-filens programbeskrivning. Annars identifieras inte det nya paketet som en uppdatering.

Obs! *När du publicerar ett nytt AIR-program med AIR 1.5.3 eller senare bör du inte ange ett utgivar-ID.*

Terminologi

I det här avsnittet finns en ordlista med några nyckeltermer som du bör förstå när du bestämmer hur du vill signera programmet för offentlig distribution.

Term	Beskrivning
Certifikatutfärdare	En enhet som fungerar som en betrodd tredje part och certifierar identiteten för den som äger en offentlig nyckel. En certifikatutfärdare utfärdar digitala certifikat, signerade med sin egen privata nyckel, som attesterar att den har verifierat certifikatinnehavarens identitet.
CPS (Certificate Practice Statement)	Reglerar metoder och principer för hur certifikatutfärdaren utfärdar och verifierar certifikat. CPS utgör en del av avtalet mellan certifikatutfärdaren och dess abonnenter och kunder. Det reglerar också principerna för den identitetskontroll och säkerhetsnivå som erbjuds i certifikaten.
Lista över återkallade certifikat	En lista med utfärdade certifikat som har återkallats och du inte längre bör lita på. AIR kontrollerar i listan med återkallade certifikat när AIR-programmet signeras och, om ingen tidstämpel finns, återigen när programmet installeras.
Certifikatkedja	En certifikatkedja är en sekvens med certifikat där varje certifikat i kedjan har signerats av nästa certifikat.
Digitalt certifikat	Ett digitalt dokument som innehåller information om ägarens identitet, ägarens offentliga nyckel och identiteten för själva certifikatet. Ett certifikat som har utfärdats av en certifikatutfärdare är självt signerat av ett certifikat som tillhör certifikatutfärdaren.
Digital signatur	Ett krypterat meddelande eller en sammanfattning kan bara dekrypteras med den offentliga nyckel som ingår i paret med offentlig/privat nyckel. I en PKI innehåller en digital signatur ett eller flera digitala certifikat som kan spåras till certifikatutfärdaren. En digital signatur kan användas för att verifiera att ett meddelande (eller datorfil) inte har ändrats sedan den signerades (inom de säkerhetsgränser som anges i den kryptografiska algoritm som används) och, förutsatt att du litar på certifikatutfärdaren, signerarens identitet.
Nyckelbehållare	En databas som innehåller digitala certifikat och, i vissa fall, relaterade privata nycklar.
Java Cryptography Architecture (JCA)	En utökningsbar arkitektur för hantering och åtkomst av nyckelbehållare. Mer information finns i Java Cryptography Architecture Reference Guide.
PKCS #11	Gränssnittsstandard för kryptografisk token från RSA Laboratories. En maskinvarutokenbaserad nyckelbehållare.
PKCS #12	Syntaxstandard för personligt informationsutbyte från RSA Laboratories. En filbaserad nyckelbehållare som vanligtvis innehåller en privat nyckel och tillhörande digitalt certifikat.
Privat nyckel	Den privata delen i ett asymmetriskt kryptografiskt system som består av en offentlig och en privat nyckel. Den privata nyckeln måste hemlighållas och bör aldrig överföras via ett nätverk. Digitalt signerade meddelanden krypteras med den privata nyckeln av signeraren.
Offentlig nyckel	Den offentliga delen i ett assymetriskt kryptografiskt system som består av en offentlig och en privat nyckel. Den offentliga nyckeln är öppet tillgänglig och används för att dekryptera meddelanden som har krypterats med den privata nyckeln.
Public Key Infrastructure (PKI)	Ett förtroendesystem där certifikatutfärdare bekräftar identiteten för ägare av offentliga nycklar. Klienter i nätverket förlitar sig på digitala certifikat som utfärdas av en betrodd certifikatutfärdare när de kontrollerar identiteten för den som har signerat ett digitalt meddelande (eller fil).
Tidstämpel	Ett digitalt signerat datum som innehåller datumet och klockslaget när en händelse inträffade. ADT kan innehålla en tidstämpel från en RFC 3161-kompatibel tidsserver i ett AIR-paket. När det finns en tidstämpel använder AIR den för att kontrollera att ett certifikat är giltigt vid tiden för signering. Det gör att ett AIR-program kan installeras efter att certifikatet har upphört att gälla.
Tidstämpelutfärdare	En enhet som utfärdar tidstämplar. För att AIR ska känna igen tidstämpeln måste den följa RFC 3161 och tidstämpelsignaturen måste ingå i en kedja till ett betrott rotcertifikat på installationsdatorn.

iOS-certifikat

De kodsigneringscertifikat som utfärdas av Apple används för att signera iOS-program, bland annat sådana som utvecklas med Adobe AIR. Du måste använda en signatur med ett utvecklingscertifikat från Apple för att installera ett program på testenheterna. Du måste använda en signatur med ett distributionscertifikat för att distribuera det färdiga programmet.

För att ADT ska signera ett program krävs att ADT har tillgång till både kodsigneringscertifikatet och till den associerade privata nyckeln. Själva certifikatfilen innehåller inte den privata nyckeln. Du måste skapa en nyckelbehållare som en Personal Information Exchange-fil (.p12 eller .pfx), som innehåller både certifikatet och den privata nyckeln. Läs mer i ”Konvertera ett utvecklarcertifikat till en P12-fil” på sidan 194.

Skapa en CSR-fil

Om du vill erhålla ett utvecklarcertifikat skapar du en CSR-fil (Certificate Signing Request), som du skickar in på webbplatsen Apple iOS Provisioning Portal.

I CSR-processen skapas ett nyckelpar med en offentlig och en privat nyckel. Den privata nyckeln finns bara på din dator. Du skickar CSR-filen med den offentliga nyckeln och din identifieringsinformation till Apple, som fungerar som certifikatutfärdare. Apple signerar ditt certifikat med deras eget WWDR-certifikat (World Wide Developer Relations).

Skapa en CSR-fil i Mac OS

På en dator med Mac OS kan du använda programmet Nyckelhanterare för att skapa en CSR-fil. Programmet Nyckelhanterare finns i underkatalogen till Verktyg i Program-katalogen. Anvisningar om hur du skapar CSR-filen finns på webbplatsen Apple iOS Provisioning Portal.

Skapa en CSR-fil i Windows

För Windows-utvecklare kan det vara lättast att få iPhone-utvecklarcertifikatet på en Mac-dator. Men det är även möjligt att få ett certifikat på en Windows-dator. Först skapar du en CSR-fil (certificate signing request) med OpenSSL:

1. Installera OpenSSL på din Windows-dator. (Gå till http://www.openssl.org/related/binaries.html.)

   Du kanske även måste installera Visual C++ 2008 Redistributable-filer, som visas på hämtningssidan för OpenSSL. (Du behöver *inte* ha Visual C++ installerad på datorn.)

2. Öppna kommandoprompten i Windows och gå till OpenSSL bin-katalogen (till exempel c:\OpenSSL\bin\).

3. Skapa den personliga nyckeln genom att ange följande i kommandoraden:

   ```
   openssl genrsa -out mykey.key 2048
   ```

   Spara den här personliga nyckelfilen. Du kommer att behöva den senare.

   När du använder OpenSSL ska du inte ignorera felmeddelanden. Om du får ett felmeddelande i OpenSSL kan programmet ändå generera filer. Dessa filer kommer kanske inte att vara användbara. Om du får ett felmeddelande ska du kontrollera syntaxen och köra om kommandot.

4. Skapa CSR-filen genom att ange följande i kommandoraden:

   ```
   openssl req -new -key mykey.key -out CertificateSigningRequest.certSigningRequest  -subj
   "/emailAddress=yourAddress@example.com, CN=John Doe, C=US"
   ```

   Byt ut värden för e-postadress, CN (certifikatnamn) och C (land) med dina egna värden.

5. Överför CSR-filen till Apple på webbplatsen för iPhone-utvecklare. (Se även "Ansöka om ett iPhone-utvecklarcertifikat och skapa en provisioneringsprofil".)

Konvertera ett utvecklarcertifikat till en P12-fil

För att skapa en P12-nyckelbehållare måste du kombinera ditt utvecklarcertifikat från Apple och den associerade privata nyckeln i en och samma fil. Exakt hur du skapar nyckelbehållaren beror på vilken metod du använde för att generera den ursprungliga CSR-filen och på var den privata nyckeln lagras.

Konvertera iPhone-utvecklarcertifikatet till en P12-fil i Mac OS

När du har hämtat iPhone-certifikatet från Apple exporterar du det till ett P12-nyckelbehållarformat. Så här gör du i Mac® OS:

1. Öppna programmet Nyckelhanterare (i mappen Program/Verktygsprogram).
2. Om du inte redan lagt till certifikatet till nyckelhanteraren väljer du Arkiv > Importera. Navigera sedan till certifikatfilen (.cer-filen) som du fick från Apple.
3. Välj nyckelkategorin i Nyckelhanterare.
4. Välj den personliga nyckeln som är associerad med ditt iPhone Development Certificate.

   Den personliga nyckeln identifieras av iPhone-utvecklaren: <Förnamn> <Efternamn>, öppet certifikat som är kopplat till det.
5. Kommando-klicka på iPhone Developer-certifikatet och välj *Exportera "iPhone-utvecklare: Namn..."*.
6. Spara nyckelbehållaren i Personal Information Exchange-format (.p12).
7. Du uppmanas att skapa ett lösenord, som sedan används när du använder nyckelbehållaren för att signera program eller när du överför nyckeln och certifikatet i den här nyckelbehållaren till en annan nyckelbehållare.

Konvertera ett Apple-utvecklarcertifikat till en P12-fil i Windows

För att kunna utveckla AIR for iOS-program måste du använda en P12-certifikatfil. Du skapar detta certifikat baserat på filen med Apple iPhone-utvecklarcertifikatet som du får från Apple.

1. Konvertera filen för utvecklarcertifikatet som du får från Apple till en PEM-certifikatfil. Kör följande kommandoradssats från bin-katalogen i OpenSSL:

   ```
   openssl x509 -in developer_identity.cer -inform DER -out developer_identity.pem -outform PEM
   ```
2. Om du använder den personliga nyckeln från nyckelringen på en Mac-dator konverterar du den till en PEM-nyckel:

   ```
   openssl pkcs12 -nocerts -in mykey.p12 -out mykey.pem
   ```
3. Du kan nu skapa en giltig P12-fil baserat på nyckeln och på PEM-versionen av iPhone-utvecklarcertifikatet:

   ```
   openssl pkcs12 -export -inkey mykey.key -in developer_identity.pem -out iphone_dev.p12
   ```

   Om du använder en nyckel från nyckelringen i Mac OS använder du PEM-versionen som du skapade i föregående steg. Annars använder du OpenSSL-nyckeln som du skapade tidigare (i Windows).

Skapa en osignerad mellanliggande AIR-fil med ADT

Använd kommandot `-prepare` om du vill skapa en osignerad mellanliggande AIR-fil. En mellanliggande AIR-fil måste signeras med ADT-kommandot `-sign` om vill skapa en giltig AIR-installationsfil.

Kommandot `-prepare` använder samma flaggor och parametrar som kommandot `-package` (med undantag för signeringsalternativen). Den enda skillnaden är att utdatafilen inte signeras. Den mellanliggande filen skapas med följande filnamnstillägg: `airi`.

Om du vill signera en mellanliggande AIR-fil använder du ADT-kommandot `-sign`. (Se ”ADT-kommandot prepare” på sidan 169.)

Exempel på ADT-kommandot -prepare

```
adt -prepare unsignedMyApp.airi myApp.xml myApp.swf components.swc
```

Signera en mellanliggande AIR-fil med ADT

Om du vill signera en mellanliggande AIR-fil med ADT använder du kommandot `-sign`. Signeringskommandot fungerar bara med mellanliggande AIR-filer (filnamnstillägget `airi`). En AIR-fil kan inte signeras en andra gång.

Om du vill skapa en mellanliggande AIR-fil använder du ADT-kommandot `-prepare`. (Se ”ADT-kommandot prepare” på sidan 169.)

Signera en AIRI-fil

❖ Använd ADT-kommandot `-sign` med följande syntax:

```
adt -sign SIGNING_OPTIONS airi_file air_file
```

SIGNING_OPTIONS Signeringsalternativen identifierar den privata nyckeln och det certifikat som används för att signera AIR-filen. Dessa alternativ beskrivs i ”ADT-kodsigneringsalternativ” på sidan 176.

airi_file Sökvägen till den osignerade mellanliggande AIR-filen som ska signeras.

air_file Namnet på AIR-filen som ska skapas.

Exempel på ADT-kommandot -sign

```
adt -sign -storetype pkcs12 -keystore cert.p12 unsignedMyApp.airi myApp.air
```

Du hittar mer information i ”ADT-kommandot sign” på sidan 169.

Signera en uppdaterad version av ett AIR-program

Varje gång du skapar en uppdaterad version av ett befintligt AIR-program signerar du det uppdaterade programmet. I bästa fall kan du använda samma certifikat, som du använde för att signera den tidigare versionen, för att signera den uppdaterade versionen. I så fall går signeringen till på samma sätt som när du signerar programmet för första gången.

Om det certifikat som användes för att signera den tidigare versionen av programmet har upphört att gälla och förnyats eller ersatts, kan du använda det förnyade eller nya certifikatet för att signera den uppdaterade versionen. Det gör du genom att signera programmet med det nya certifikatet *och* använda en flyttningssignatur med originalcertifikatet. Flyttningssignaturen bekräftar att det ursprungliga certifikatets ägare har publicerat uppdateringen.

Tänk på följande innan du använder en flyttningssignatur:

- För att kunna använda en flyttningssignatur måste det ursprungliga certifikatet fortfarande vara giltigt eller ha upphört att gälla under de senaste 365 dagarna. Längden på den här fristen kan komma att ändras i framtiden.

  Obs! *Fram till AIR 2.6 var fristen 180 dagar.*

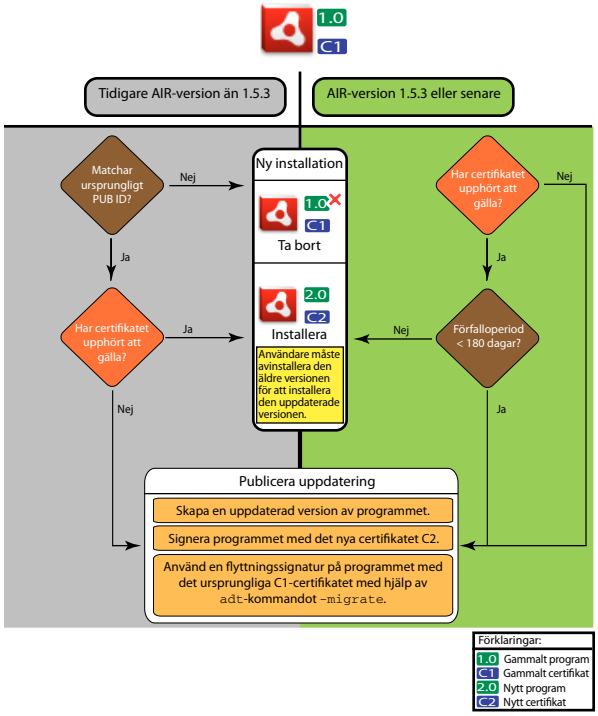
- Du kan inte använda en flyttningssignatur om certifikatet har upphört att gälla och den 365-dagar långa fristen har förflutit. I så fall måste användarna avinstallera den befintliga versionen innan de kan installera den uppdaterade.

- Fristen på 365 dagar gäller endast för program där AIR-version 1.5.3 eller senare anges i namnutrymmet för programbeskrivningen.

Viktigt! *Att signera uppdateringar med flyttningssignaturer från utgångna certifikat är en tillfällig lösning. Om du vill ha en komplett lösning skapar du ett standardiserat signeringsarbetsflöde för att hantera distribution av programuppdateringar. Signera till exempel varje uppdatering med det senaste certifikatet och använd ett flyttningscertifikat tillsammans med det certifikat som användes för att signera den förra uppdateringen (i förekommande fall). Överför varje uppdatering till en egen webbadress (URL) från vilken användare kan hämta programmet. Mer information finns i "Signeringsarbetsflöde för uppdateringar" på sidan 255.*

Följande tabell och bilder summerar arbetsflödet för flyttningssignaturer:

Scenario	**Status för ursprungligt certifikat**	**Utvecklaråtgärd**	**Användaråtgärd**
Program baserat på Adobe AIR version 1.5.3 eller senare	Giltigt	Publicera den senaste versionen av AIR-programmet	Ingen åtgärd krävs Programmet uppdateras automatiskt
	Upphört att gälla, men inom 365-dagarsfristen	Signera programmet med det nya certifikatet. Använd en flyttningssignatur med det certifikat som upphört att gälla.	Ingen åtgärd krävs Programmet uppdateras automatiskt
	Har upphört att gälla och fristen har gått ut	Du kan inte använda flyttningssignaturen på AIR-programuppdateringen. Du måste i stället publicera en annan version av AIR-programmet med ett nytt certifikat. Användare kan installera den nya versionen när de har avinstallerat den befintliga versionen av AIR-programmet.	Avinstallera den aktuella versionen av AIR-programmet och installera den senaste versionen
• Program baserat på Adobe AIR version 1.5.2 eller tidigare • Utgivar-ID:t i uppdateringens programbeskrivning matchar utgivar-ID:t för den tidigare versionen.	Giltigt	Publicera den senaste versionen av AIR-programmet	Ingen åtgärd krävs Programmet uppdateras automatiskt
	Har upphört att gälla och fristen har gått ut	Du kan inte använda flyttningssignaturen på AIR-programuppdateringen. Du måste i stället publicera en annan version av AIR-programmet med ett nytt certifikat. Användare kan installera den nya versionen när de har avinstallerat den befintliga versionen av AIR-programmet.	Avinstallera den aktuella versionen av AIR-programmet och installera den senaste versionen
• Program baserat på Adobe AIR version 1.5.2 eller tidigare • Utgivar-ID:t i uppdateringens programbeskrivning **matchar inte** utgivar-ID:t för den tidigare versionen.	Alla	Signera AIR-programmet med ett giltigt certifikat och publicera den senaste versionen av AIR-programmet.	Avinstallera den aktuella versionen av AIR-programmet och installera den senaste versionen



Signeringsarbetsflöde för uppdateringar

Flytta ett AIR-program till att använda ett nytt certifikat

Så här flyttar du ett AIR-program till ett nytt certifikat medan du uppdaterar programmet:

1. Skapa en uppdatering av programmet.
2. Paketera och signera AIR-uppdateringsfilen med det **nya** certifikatet.
3. Signera AIR-filen igen med det **ursprungliga** certifikatet med hjälp av kommandot `-migrate`

En AIR-fil som signerats med kommandot `-migrate` kan också användas för att installera en ny version av programmet, inte bara för att uppdatera tidigare versioner som signerats med det gamla certifikatet.

Obs! *När du uppdaterar ett program som publicerats för en tidigare version av AIR än 1.5.3 anger du det ursprungliga utgivar-ID:t i programbeskrivningen. Annars måste användare av programmet avinstallera den tidigare versionen innan de installerar uppdateringen.*

Använd ADT-kommandot `-migrate` med följande syntax:

```
adt -migrate SIGNING_OPTIONS air_file_in air_file_out
```

- **SIGNING_OPTIONS** Signeringsalternativen identifierar den privata nyckeln och det certifikat som används för att signera AIR-filen. Dessa alternativ måste identifiera det **ursprungliga** signeringscertifikatet och beskrivs i ”ADT-kodsigneringsalternativ” på sidan 176.
- **air_file_in** AIR-filen för uppdateringen, signerad med det **nya** certifikatet.
- **air_file_out** AIR-filen som ska skapas.

Obs! *Namnen på de filer som används för in- och utdata i AIR måste vara olika.*

I följande exempel visas hur ADT anropas med flaggan `-migrate` för att använda en flyttningssignatur på en uppdaterad version av ett AIR-program:

```
adt -migrate -storetype pkcs12 -keystore cert.p12 myAppIn.air myApp.air
```

Obs! *Kommandot* `-migrate` *lades till i ADT i AIR 1.1-utgåvan.*

Flytta ett AIR-program med systemspecifikt installationsprogram så att det använder ett nytt certifikat

Ett AIR-program som publiceras som ett systemspecifikt installationsprogram (till exempel ett program som använder det systemspecifika tillägget api) kan inte signeras med ADT-kommandot `-migrate`, eftersom det är ett plattformsberoende program, inte en .air-fil. Om du vill flytta ett AIR-program som publicerats som ett systemspecifikt tillägg så att det använder ett nytt certifikat gör du i stället så här:

1. Skapa en uppdatering av programmet.
2. Kontrollera att taggen `<supportedProfiles>` i programbeskrivningsfilen (app.xml) innehåller både profilen desktop och profilen extendedDesktop (eller ta bort taggen `<supportedProfiles>` från programbeskrivningen).
3. Paketera och signera det uppdaterade programmet **som en .air-fil** med ADT-kommandot `-package` och det **nya** certifikatet.
4. Använd flyttningscertifikatet på .air-filen med ADT-kommandot `-migrate` och det **ursprungliga** certifikatet (enligt beskrivningen tidigare i ”Flytta ett AIR-program till att använda ett nytt certifikat” på sidan 197).
5. Paketera .air-filen som ett systemspecifikt installationsprogram med ADT-kommandot `-package` och flaggan `-target native`. Eftersom programmet redan har signerats anger du inget signeringscertifikat i det här steget.

I följande exempel beskrivs steg 3–5 i processen. Koden anropar ADT med kommandot `-package`, anropar ADT med kommandot `-migrate` och anropar sedan ADT igen med kommandot `-package` för att paketera en uppdaterad version av ett AIR-program som ett systemspecifikt installationsprogram:

```
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
```

Flytta ett AIR-program som använder ett systemspecifikt tillägg så att det använder ett nytt certifikat

Ett AIR-program som använder ett systemspecifikt tillägg kan inte signeras med ADT-kommandot `-migrate`. Det kan inte heller flyttas på samma sätt som ett AIR-program som använder ett systemspecifikt installationsprogram, eftersom det inte kan publiceras som en mellanliggande .air-fil. Om du vill flytta ett AIR-program som använder ett systemspecifikt tillägg så att det använder ett nytt certifikat gör du i stället så här:

1. Skapa en uppdatering av programmet.
2. Paketera och signera det uppdaterade systemspecifika installationsprogrammet med ADT-kommandot `-package`. Paketera programmet med det **nya** certifikatet och inkludera flaggan `-migrate` med det **ursprungliga** certifikatet.

Använd följande syntax för att anropa ADT-kommandot `-package` med flaggan `-migrate`:

```
adt -package AIR_SIGNING_OPTIONS -migrate MIGRATION_SIGNING_OPTIONS -target package_type
NATIVE_SIGNING_OPTIONS output app_descriptor FILE_OPTIONS
```

- **AIR_SIGNING_OPTIONS** Signeringsalternativen identifierar den privata nyckeln och det certifikat som används för att signera AIR-filen. Dessa alternativ identifierar det **nya** signeringscertifikatet och beskrivs i ”ADT-kodsigneringsalternativ” på sidan 176.

- **MIGRATION_SIGNING_OPTIONS** Signeringsalternativen identifierar den privata nyckeln och det certifikat som används för att signera AIR-filen. Dessa alternativ identifierar det **ursprungliga** signeringscertifikatet och beskrivs i ”ADT-kodsigneringsalternativ” på sidan 176.
- De andra alternativen är samma alternativ som används för att paketera ett AIR-program med systemspecifikt installationsprogram och beskrivs i ”ADT-kommandot package” på sidan 163.

Följande exempel visar hur ADT anropas med kommandot `-package` och flaggan `-migrate` för att paketera en uppdaterad version av ett AIR-program som använder ett systemspecifikt tillägg och använda en flyttningssignatur på uppdateringen:

```
adt -package -storetype pkcs12 -keystore new_cert.p12 -migrate -storetype pkcs12 -keystore
original_cert.p12 -target native myApp.exe myApp.xml myApp.swf
```

Obs! *Flaggan `-migrate` i kommandot `-package` finns i ADT i AIR 3.6 och senare versioner.*

Skapa ett självsignerat certifikat med ADT

Du kan använda självsignerade certifikat för att skapa en giltig AIR-installationsfil. Självsignerade certifikat ger emellertid endast en begränsad säkerhet för användarna. Självsignerade certifikats äkthet kan inte kontrolleras. När en självsignerad AIR-fil installeras visas utgivarinformationen för användarna som Okänd. Ett certifikat som skapas av ADT gäller i fem år.

Om du skapar en uppdatering för ett AIR-program som har signerats med ett självsignerat certifikat, måste du använda samma certifikat för att signera både de ursprungliga AIR-filerna och AIR-uppdateringsfilerna. De certifikat som ADT skapar är alltid unika, även om samma parametrar används. Om du vill självsignera uppdateringar med ett ADT-skapat certifikat ska du därför spara det ursprungliga certifikatet på en säker plats. Du kan heller inte skapa en uppdaterad AIR-fil efter att det ursprungliga ADT-skapade certifikatet har upphört att gälla. (Du kan publicera nya program med ett annat certifikat, men inte nya versioner av samma program.)

Viktigt! *På grund av begränsningarna i självsignerade certifikat rekommenderar Adobe att du använder ett kommersiellt certifikat från en betrodd certifikatutfärdare om du vill signera AIR-program för allmänheten.*

Det certifikat och den tillhörande privata nyckel som skapas av ADT lagras i en nyckelfil av typen PKCS12. Lösenordet som anges ställs in i själva nyckeln, inte i nyckelbehållaren.

Exempel på hur certifikat skapas

```
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
```

Om du vill använda dessa certifikat för att signera AIR-filer ska du använda följande alternativ med ADT-kommandot `-package` eller `-prepare`:

```
-storetype pkcs12 -keystore newcert.p12 -storepass 39#wnetx3tl
-storetype pkcs12 -keystore SigningCert.p12 -storepass 39#wnetx3tl
```

Obs! *I Java-versionerna 1.5 och senare går det inte att använda hög-ASCII-tecken i lösenord för att skydda PKCS12-certifikatfiler. Använd endast reguljära ASCII-tecken i lösenordet.*

Kapitel 14: AIR-programbeskrivningsfiler

Alla AIR-program måste ha en programbeskrivningsfil. Programbeskrivningsfilen är ett XML-dokument som definierar programmets grundläggande egenskaper.

I många utvecklingsmiljöer med stöd för AIR genereras automatiskt en programbeskrivning när du skapar ett projekt. I andra fall måste du själv skapa en egen beskrivningsfil. Ett exempel på en beskrivningsfil, `descriptor-sample.xml`, hittar du i katalogen `samples` både i AIR och i Flex SDKs.

Valfritt filnamn kan användas för programbeskrivningsfilen. När du paketerar programmet ändras namnet på programbeskrivningsfilen till `application.xml` och placeras i en särskild katalog i AIR-paketet.

Exempel på programbeskrivning
Följande programbeskrivningsfil anger de grundläggande egenskaper som används av de flesta AIR-program:

```
<?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>
```

Om en HTML-fil i stället för en SWF-fil används som programmets rotinnehåll är endast elementet `<content>` annorlunda:

```
<content>
    HelloWorld.html
</content>
```

Ändringar av programbeskrivningsfilen

AIR-programbeskrivningen har ändrats i följande versioner av AIR.

Ändringar i AIR 1.1-programbeskrivningen

Programmets `name`- och `description`-element kan lokaliseras med hjälp av `text`-elementet.

Ändringar i AIR 1.5-programbeskrivningsfilen

`contentType` blev ett nödvändigt underordnat element till `fileType`.

Ändringar i AIR 1.5.3-programbeskrivningsfilen

Elementet `publisherID` har lagts till så att program ska kunna ange ett värde för utgivar-ID.

Ändringar i AIR 2.0-programbeskrivningsfilen

Tillagt:

- `aspectRatio`
- `autoOrients`
- `fullScreen`
- `image29x29` (se imageNxN)
- `image57x57`
- `image72x72`
- `image512x512`
- `iPhone`
- `renderMode`
- `supportedProfiles`

Ändringar i AIR 2.5-programbeskrivningsfilen

Borttaget: `version`

Tillagt:

- `android`
- `extensionID`
- `extensions`
- `image36x36` (se imageNxN)
- `manifestAdditions`
- `versionLabel`
- `versionNumber`

Ändringar i AIR 2.6-programbeskrivningsfilen

Tillagt:

- `image114x114` (se `imageNxN`)
- `requestedDisplayResolution`

- `softKeyboardBehavior`

Ändringar i AIR 3.0-programbeskrivningsfilen

Tillagt:

- `colorDepth`
- *direct* som ett giltigt värde för `renderMode`
- renderMode ignoreras inte längre på skrivbordsplattformar
- Android-elementet `<uses-sdk>` kan anges. (Detta var inte tillåtet tidigare.)

Ändringar i AIR 3.1-programbeskrivningsfilen

Tillagt:

- ”Entitlements” på sidan 214

Ändringar i AIR 3.2-programbeskrivningsfilen

Tillagt:

- `depthAndStencil`
- `supportedLanguages`

Ändringar i AIR 3.3-programbeskrivningsfilen

Tillagt:

- `aspectRatio` innehåller nu alternativet `ANY`.

Ändringar i AIR 3.4-programbeskrivningsfilen

Tillagt:

- `image50x50` (se ”imageNxN” på sidan 222)
- `image58x58` (se ”imageNxN” på sidan 222)
- `image100x100` (se ”imageNxN” på sidan 222)
- `image1024x1024` (se ”imageNxN” på sidan 222)

Ändringar i AIR 3.6-programbeskrivningsfilen

Tillagt:

- ”requestedDisplayResolution” på sidan 232 i elementet ”iPhone” på sidan 226 inkluderar nu attributet `excludeDevices`, som används för att ange vilka iOS-målenheter som ska använda hög upplösning eller standardupplösning.
- Det nya elementet ”requestedDisplayResolution” på sidan 232 i ”initialWindow” på sidan 224 anger om hög upplösning eller standardupplösning ska användas på skrivbordsplattformar som Macintosh-datorer med högupplösta skärmar.

Ändringar i AIR 3.7-programbeskrivningsfilen

Tillagt:

- Elementet ”iPhone” på sidan 226 innehåller nu ett ”externalSwfs” på sidan 216-element, som du använder för att skapa en lista med SWF-filer som ska läsas under körningen.
- Elementet ”iPhone” på sidan 226 innehåller nu ett ”forceCPURenderModeForDevices” på sidan 219-element som gör att du kan framtvinga CPU-återgivningsläge för vissa enheter.

Struktur för programbeskrivningsfil

Programbeskrivningsfilen är ett XML-dokument med följande struktur:

```
<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>
    <”supportedLanguages” på sidan 234>...</”supportedLanguages” på sidan 234>
    <supportedProfiles>...</supportedProfiles>
    <versionNumber>...</versionNumber>
    <versionLabel>...</versionLabel>
</application>
```

Element i AIR-programbeskrivningsfilen

Följande lista med element beskriver vart och ett av de giltiga elementen i en AIR-programbeskrivningsfil.

allowBrowserInvocation

Adobe AIR 1.0 och senare (valfritt)

Används för att webbläsar-API:t för AIR ska kunna identifiera och starta programmet.

Om du anger att det här värdet ska vara `true` måste du ta hänsyn till säkerhetsaspekterna. Dessa beskrivs i Anropa ett AIR-program från webbläsaren (for ActionScript developers) och Invoking an AIR application from the browser (för HTML-utvecklare).

Du hittar mer information i ”Starta ett installerat AIR-program från webbläsaren” på sidan 251.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll

`true` eller `false` (standard)

Exempel

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

android

Adobe AIR 2.5 och senare (valfritt)

Använd för att lägga till element i Androids manifestfil. AIR skapar filen AndroidManifest.xml för varje APK-paket. Du kan använda elementet android i AIR-programbeskrivningsfilen för att lägga till ytterligare element i den. Ignoreras på alla plattformar utom Android.

Överordnat element:”application” på sidan 205

Underordnade element:

- ”colorDepth” på sidan 209
- ”manifestAdditions” på sidan 227

Innehåll

Element som definierar de Android-specifika egenskaper som ska läggas till i Androids manifestfil för programmet.

Exempel

```
<android>
    <manifestAdditions>
        ...
    </manifestAdditions>
</android>
```

Fler hjälpavsnitt

”Android-inställningar” på sidan 73

The AndroidManifest.xml File (Filen AndroidManifest.xml)

application

Adobe AIR 1.0 och senare (krävs)

Rotelementet i en AIR-programbeskrivningsfil.

Överordnat element: inga

Underordnade element:

- ”allowBrowserInvocation” på sidan 204
- ”android” på sidan 205
- ”copyright” på sidan 211
- ”customUpdateUI” på sidan 211
- ”description” på sidan 213

- ”extensions” på sidan 215
- ”filename” på sidan 216
- ”fileTypes” på sidan 218
- ”icon” på sidan 221
- ”id” på sidan 221
- ”initialWindow” på sidan 224
- ”installFolder” på sidan 225
- ”iPhone” på sidan 226
- ”name” på sidan 229
- ”programMenuFolder” på sidan 231
- ”publisherID” på sidan 231
- ”supportedLanguages” på sidan 234
- ”supportedProfiles” på sidan 234
- ”version” på sidan 237
- ”versionLabel” på sidan 237
- ”versionNumber” på sidan 238

Attribut

minimumPatchLevel – Den lägsta korrigeringsnivå för AIR-miljön som krävs för det här programmet.

xmlns – Attributet för XML-namnutrymme anger den version av AIR-miljön som krävs för programmet.

Namnutrymmet ändras vid varje större versionsändring av AIR (men inte vid mindre korrigeringar). Den sista delen av namnutrymmet, till exempel ”3.0”, visar vilken version av miljön som krävs för programmet.

xmlns-värdena för de huvudsakliga AIR-versionerna är:

```
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"
```

I SWF-baserade program avgör AIR-runtimeversionen som anges i programbeskrivningsfilen vilken den högsta tillåtna SWF-versionen är som kan läsas in som ursprungligt innehåll för programmet. Program som anger AIR 1.0 eller AIR 1.1 kan bara använda SWF9-filer (Flash Player 9) som ursprungligt innehåll. Detta gäller även när AIR 2 används. I program som avser AIR 1.5 (eller senare) kan antingen SWF9- eller SWF10-filer användas (Flash Player 10) som ursprungligt innehåll.

SWF-versionen avgör vilka versioner av AIR- och Flash Player-API:er som är tillgängliga. Om en SWF9-fil används som ursprungligt innehåll i ett AIR 1.5-program har det programmet bara tillgång till API:er för AIR 1.1 och Flash Player 9. Dessutom kommer ändringar av beteendet i befintliga API:er i AIR 2.0 eller Flash Player 10.1 inte att tillämpas. (Viktiga säkerhetsrelaterade ändringar i API:er är undantag från regeln och kommer att tillämpas retroaktivt i befintliga eller kommande korrigeringsuppdateringar av runtime-modulen.)

I HTML-baserade program avgör runtime-versionen som anges i programbeskrivningsfilen vilken version av AIR- och Flash Player-API:er som är tillgänglig för programmet. HTML-, CSS- och JavaScript-beteenden bestäms alltid av vilken version av Webkit som används i den installerade AIR-runtime-modulen, inte av programbeskrivningsfilen.

När ett AIR-program läser in SWF-innehåll avgör sättet som innehållet läses in på vilka versioner av API:er i AIR och Flash Player som är tillgängliga. Ibland bestäms den aktuella versionen av namnutrymmet för programbeskrivningen, ibland bestäms den av versionen av innehållet som läses in och ibland av versionen av det inlästa innehållet. Följande tabell visar hur API-versionen bestäms av inläsningsmetoden:

Hur innehållet läses in	**Hur API-versionen bestäms**
Ursprungligt innehåll, SWF-baserat program	SWF-version av den **inlästa** filen
Ursprungligt innehåll, HTML-baserat program	Namnutrymme för programbeskrivningsfilen
SWF inläst av SWF-innehåll	Version av innehållet som **läser in**
SWF-bibliotek inläst av HTML-innehåll med <script>-taggen	Namnutrymme för programbeskrivningsfilen
SWF inläst av HTML-innehåll med AIR- eller Flash Player-API:er (t. ex. flash.display.Loader)	Namnutrymme för programbeskrivningsfilen
SWF inläst av HTML-innehåll med taggarna <object> eller <embed> (eller motsvarande JavaScript-API:er)	SWF-version för den **inlästa** filen

När du läser in en SWF-fil som har en annan version än innehållet som läser in, kan du stöta på två problem:

- Läsa in en nyare SWF-version med en äldre SWF-version – Referenser till API:er tillagda i senare versioner av AIR och Flash Player i det inlästa innehållet kommer inte att lösas.
- Läsa in en äldre SWF-version med en nyare SWF-version – API:er ändrade i de nyare versionerna av AIR och Flash Player kan uppträda på ett sätt som inte är kompatibelt med det inlästa innehållet.

Innehåll

Elementet application innehåller underordnade element, som definierar egenskaperna för ett AIR-program.

Exempel

```
<?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 och senare, iOS och Android – (valfritt)

Anger programmets proportioner.

Om det inte anges öppnas programmet med enhetens ”naturliga” proportioner och orientering. Den naturliga orienteringen varierar från enhet till enhet. Vanligtvis är orienteringen stående på enheter med små skärmar, som telefoner. På vissa enheter, som iPad-plattor, öppnas programmet i den aktuella orienteringen. I AIR 3.3 och senare gäller detta för hela programmet, inte bara den första visningen.

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll

`portrait`, `landscape` eller `any`

Exempel

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

autoOrients

Adobe AIR 2.0 och senare, iOS och Android – (valfritt)

Anger om programmet innehållets automatiskt ska orienteras om när enhetens fysiska orientering ändras. Mer information finns i Scenorientering.

Om du använder automatisk orientering kan det vara en god idé att ange egenskaperna `align` och `scaleMode` för scenen enligt följande:

```
stage.align = StageAlign.TOP_LEFT;
stage.scaleMode = StageScaleMode.NO_SCALE;
```

Med de här inställningarna kan programmet roteras runt det övre, vänstra hörnet, vilket hindrar programinnehållet från att skalas automatiskt. Andra skalningslägen anpassar visserligen innehållet så att det passar dimensionerna på den roterade scenen, men de klipper även av, förvränger eller krymper innehållet avsevärt. Du får nästan alltid bättre resultat om du ritar om innehållet eller layouten själv.

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll

`true` eller `false` (standard)

Exempel

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

colorDepth

Adobe AIR 3 och senare (valfritt)

Anger om 16-bitars eller 32-bitars färg ska användas.

Om du använder 16-bitars färg kan återgivningsprestandan förbättras, men på bekostnad av färgåtergivningen. Före AIR 3 används alltid 16-bitars färg för Android. I AIR 3 används 32-bitars färg som standard.

Obs! *Om klassen StageVideo används i programmet måste du använda 32-bitars färg.*

Överordnat element:”android” på sidan 205

Underordnade element: inga

Innehåll
Ett av följande värden:

- 16 bitar
- 32 bitar

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

containsVideo

Anger om programmet ska innehålla videoinnehåll eller inte.

Överordnat element:”android” på sidan 205

Underordnade element: inga

Innehåll
Ett av följande värden:

- true
- false

Exempel
```
<android>
    <containsVideo>true</containsVideo>
    <manifestAdditions>...</manifestAdditions>
</android>
```

content

Adobe AIR 1.0 och senare (krävs)

Värdet som anges för `content`-elementet är URL:en för programmets huvudinnehållsfil. Det kan vara en SWF-fil eller en HTML-fil. URL:en är angiven i förhållande till roten för programmets installationsmapp. (Om ett AIR-program med ADL körs är URL:en relativ till den mapp som innehåller programbeskrivningsfilen. Du kan använda `root-dir`-parametern i ADL för att ange en annan rotkatalog.)

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll
En URL som anges i förhållande till programkatalogen. Eftersom värdet för content-elementet behandlas som en URL måste tecknen i innehållsfilens namn vara URL-kodade enligt reglerna i RFC 1738. Blanksteg måste till exempel vara kodade som `%20`.

Exempel

```
<content>TravelPlanner.swf</content>
```

contentType

Adobe AIR 1.0 till 1.1 (valfritt); AIR 1.5 och senare (krävs)

`contentType` krävs från och med AIR 1.5 (det var valfritt i AIR 1.0 och 1.1). Med hjälp av den här egenskapen kan vissa operativsystem hitta det bästa programmet för att öppna en viss fil. Värdet ska vara MIME-typen för filinnehållet. Lägg märke till att värdet ignoreras på Linux om filtypen redan är registrerad och har en tilldelad MIME-typ.

Överordnat element:”fileType” på sidan 217

Underordnade element: inga

Innehåll

MIME-typ och undertyp. Läs RFC2045 om du vill ha mer information om MIME-typer.

Exempel

```
<contentType>text/plain</contentType>
```

copyright

Adobe AIR 1.0 och senare (valfritt)

Copyrightinformationen för AIR-programmet. I Mac OS visas upphovsrättstexten i det installerade programmets Om-dialogruta. I Mac OS används också upphovsrättsinformationen i fältet NSHumanReadableCopyright i programmets Info.plist-fil.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll

En sträng som innehåller copyrightinformationen för programmet.

Exempel

```
<copyright>© 2010, Examples, Inc.All rights reserved.</copyright>
```

customUpdateUI

Adobe AIR 1.0 och senare (valfritt)

Anger om ett program tillhandahåller egna dialogrutor för uppdatering. Om värdet är `false` visas standarddialogrutor för uppdatering i AIR. Det är bara program som distribueras som AIR-filer som kan använda det inbyggda uppdateringssystemet i AIR.

Om den installerade programversionen har `customUpdateUI`-elementet angivet som `true` och användaren sedan dubbelklickar på AIR-filen för en ny version eller installerar en uppdatering av programmet med hjälp av den sömlösa installationsfunktionen, öppnas i runtime-modulen den installerade programversionen. I runtime-modulen öppnas inte standardinstallationsprogrammet för AIR. Programlogiken kan sedan avgöra hur uppdateringsåtgärden ska fortsätta. (Program-ID och utgivar-ID i AIR-filen måste överensstämma med dem som finns i det installerade programmet för att en uppgradering ska genomföras.)

Obs! *`customUpdateUI`-mekanismen används bara om programmet redan är installerat och användaren dubbelklickar på den AIR-installationsfil som innehåller en uppdatering eller installerar en uppdatering av programmet med hjälp av den sömlösa installationsfunktionen. Du kan hämta och starta en uppdatering via din egen programlogik. Du måste visa det anpassade användargränssnittet oavsett om `customUpdateUI` är `true` eller inte.*

Mer information finns i ”Uppdatera AIR-program” på sidan 253.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll

`true` eller `false` (standard)

Exempel

```
<customUpdateUI>true</customUpdateUI>
```

depthAndStencil

Adobe AIR 3.2 och senare (valfritt)

Anger att programmet kräver att djup- eller stencilbuffert ska användas. Dessa buffertar används oftast när du arbetar med 3D-innehåll. Som standard är värdet på det här elementet `false` för att inaktivera djup- och stencilbuffertar. Det här elementet krävs eftersom buffertarna måste tilldelas vid programstarten, innan något innehåll läses in.

Inställningen för det här elementet måste matcha det värde som skickas för argumentet `enableDepthAndStencil` till metoden `Context3D.configureBackBuffer()`. Om värdena inte matchar genereras ett fel i AIR.

Det här elementet används bara när `renderMode` = `direct`. Om `renderMode` inte är lika med `direct` genereras felet 118 i ADT:

```
<depthAndStencil> element unexpected for render mode cpu.  It requires "direct" render mode.
```

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll

`true` eller `false` (standard)

Exempel

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

description

Adobe AIR 1.0 och senare (valfritt)

En beskrivning av programmet, som visas i AIR-programmets installationsprogram.

Om du anger en enda textnod (inte flera textelement) använder AIR-programmets installationsprogram den här beskrivningen, oavsett systemspråk. Annars använder AIR-programmets installationsprogram den beskrivning som mest liknar språket i användargränssnittet i användarens operativsystem. Ta till exempel en installation där `description`-elementet i programbeskrivningsfilen innehåller ett värde för språkinställningen en (engelska). AIR-programmets installationsprogram använder beskrivningen en om användarens system identifierar en (engelska) som språk i användargränssnittet. Beskrivningen en används också om språket i användargränssnittet är en-US (amerikansk engelska). Om språket i användargränssnittet är en-US, och programbeskrivningsfilen definierar både namnen en-US och en-GB, använder AIR-programmets installationsprogram värdet en-US. Om programmet inte definierar någon beskrivning som matchar språket i användargränssnittet, använder AIR-programmets installationsprogram det första `description`-värde som definieras i programbeskrivningsfilen.

Mer information om utveckling av flerspråkiga program finns i ”Lokalisera AIR-program” på sidan 288.

Överordnat element:”application” på sidan 205

Underordnade element:”text” på sidan 236

Innehåll

Programbeskrivningsschemat för AIR 1.0 tillåter bara att en enda textnod anges för namnet (inte flera `text`-element).

I AIR 1.1 (eller senare) kan du ange flera språk i `description`-elementet. `xml:lang`-attributet för varje text-element anger en språkkod enligt definitionen i RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

Exempel

Beskrivning med enkel textnod:

```
<description>This is a sample AIR application.</description>
```

Beskrivning med lokaliserade textelement för engelska, franska och spanska (giltigt i AIR 1.1 och senare):

```
<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 och senare (krävs)

Filtypsbeskrivningen visas för användaren i operativsystemet. Det går inte att lokalisera filtypsbeskrivningen.

Se även: ”description” på sidan 213 som underordnat element till elementet application

Överordnat element:”fileType” på sidan 217

Underordnade element: inga

Innehåll

En sträng som beskriver filens innehåll.

Exempel

```
<description>PNG image</description>
```

embedFonts

Gör att du kan använda egna teckensnitt i StageText i AIR-programmet. Det här elementet är valfritt.

Överordnat element:”application” på sidan 205

Underordnade element:”teckensnitt” på sidan 218

Innehåll

Elementet embedFonts kan innehålla valfritt antal teckensnittselement.

Exempel

```
<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 och senare – endast iOS, valfritt

I iOS används egenskaper som kallas "entitlements" för att ge programmet åtkomst till ytterligare resurser och funktioner. Använd elementet Entitlements för att ange denna information i ett mobilt iOS-program.

Överordnat element:”iPhone” på sidan 226

Underordnade element: iOS-elementen Entitlements.plist

Innehåll

Innehåller underordnade element som anger nyckelvärdepar som används som inställningar för Entitlements.plist för programmet. Innehåll i Entitlements-elementet bör omslutas av ett CDATA-block. Mer information finns i Entitlement Key Reference i iOS Developer Library.

Exempel

```
<iphone>
...
   <Entitlements>
      <![CDATA[
         <key>aps-environment</key>
         <string>development</string>
      ]]>
   </Entitlements>
</iphone>
```

extension

Adobe AIR 1.0 och senare (krävs)

Tilläggssträngen för en filtyp.

Överordnat element:”fileType” på sidan 217

Underordnade element: inga

Innehåll
En sträng som identifierar filnamnstillägget (utan punkten, ”.”).

Exempel

```
<extension>png</extension>
```

extensionID

Adobe AIR 2.5 och senare

Anger ID:t för ett ActionScript-tillägg som används av programmet. ID:t definieras i tilläggsbeskrivningsfilen.

Överordnat element:”extensions” på sidan 215

Underordnade element: inga

Innehåll
En sträng som identifierar ActionScript-tilläggets ID.

Exempel

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

extensions

Adobe AIR 2.5 och senare (valfritt)

Identifierar de ActionScript-tillägg som används av ett program.

Överordnat element:”application” på sidan 205

Underordnade element:”extensionID” på sidan 215

Innehåll
Underordnade `extensionID`-element som innehåller ID:n för ActionScript-tillägg från tilläggsbeskrivningsfilen.

Exempel

```
<extensions>
    <extensionID>extension.first</extensionID>
    <extensionID>extension.next</extensionID>
    <extensionID>extension.last</extensionID>
</extensions>
```

externalSwfs

Adobe AIR 3.7 och senare, endast iOS (valfritt)

Anger namnet på en textfil som innehåller en lista med SWF-filer som ska konfigureras med ADT för fjärrvärdskap. Du kan minimera storleken på den första programnedladdningen genom att förpacka en delmängd av SWF-filerna som används i programmet och läsa in de återstående (resursexklusiva) externa SWF-filerna i miljön med metoden `Loader.load ()`. Om du vill använda den här funktionen måste du förpacka programmet så att ADT flyttar all ActionScript ByteCode (ABC) från de externt inlästa SWF-filerna till huvudprogrammet för SWF och lämnar en SWF-fil som endast innehåller resurser. Detta görs för att anpassa till regelverket för Apple Store som förbjuder att kod laddas ned sedan programmet har installerats.

Mer information finns i ”Minimera nedladdningstorleken genom att läsa in externa, resursexklusiva SWF-filer” på sidan 84.

Överordnat element:”iPhone” på sidan 226, ”initialWindow” på sidan 224

Underordnade element: inga

Innehåll

Namnet på en textfil som innehåller en radavgränsad lista med SWF-filer som anpassas för fjärrvärdskap.

Attribut:

Ingen.

Exempel

iOS:

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

filename

Adobe AIR 1.0 och senare (krävs)

Den sträng som används som filnamn för programmet (utan filtillägg) när programmet installeras. Programfilen startar AIR-programmet i körmiljön. Om inget `name`-värde har angetts används `filename` även som namn för installationsmappen.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll

`filename`-egenskapen kan innehålla alla Unicode-tecken (UTF-8) utom följande (eftersom de inte får användas som filnamn i olika filsystem):

Tecken	Hexadecimal kod
olika	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

`filename`-värdet får inte sluta med en punkt.

Exempel

```
<filename>MyApplication</filename>
```

fileType

Adobe AIR 1.0 och senare (valfritt)

Beskriver en enda filtyp som programmet kan registreras för.

Överordnat element:”fileTypes” på sidan 218

Underordnade element:

- ”contentType” på sidan 211
- ”description” på sidan 213
- ”extension” på sidan 215
- ”icon” på sidan 221
- ”name” på sidan 230

Innehåll

Element som beskriver en filtyp.

Exempel

```
<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 och senare (valfritt)

Med elementet `fileTypes` kan du deklarera de filtyper som ett AIR-program kan associeras med.

När ett AIR-program installeras registreras alla deklarerade filtyper med operativsystemet. Om filtyperna inte redan är associerade med ett annat program associeras de med AIR-programmet. Om du vill åsidosätta en befintlig association mellan en filtyp och ett annat program använder du metoden `NativeApplication.setAsDefaultApplication()` vid körningen (helst med användarens tillåtelse).

Obs! *Körningsmetoderna kan bara hantera associationer för de filtyper som är deklarerade i programbeskrivningen.*

Elementet `fileTypes` är valfritt.

Överordnat element:”application” på sidan 205

Underordnade element:”fileType” på sidan 217

Innehåll

Elementet `fileTypes` kan innehålla valfritt antal `fileType`-element.

Exempel

```
<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>
```

teckensnitt

Beskriver ett anpassat teckensnitt som kan användas i AIR-programmet.

Överordnat element:”embedFonts” på sidan 214

Underordnade element:”fontName” på sidan 219, ”fontPath” på sidan 219

Innehåll

Element som anger det anpassade teckensnittsnamnet och dess sökväg.

Exempel

```
<font>
   <fontPath>ttf/space age.ttf</fontPath>
   <fontName>space age</fontName>
</font>
```

fontName

Anger namnet på det anpassade teckensnittet.

Överordnat element:”teckensnitt” på sidan 218

Underordnade element: inga

Innehåll

Namnet på det anpassade teckensnittet som ska anges i StageText.fontFamily

Exempel

```
<fontName>space age</fontName>
```

fontPath

Anger platsen för den anpassade teckensnittsfilen.

Överordnat element:”teckensnitt” på sidan 218

Underordnade element: inga

Innehåll

Sökvägen till den anpassade teckensnittsfilen (med avseende på källan).

Exempel

```
<fontPath>ttf/space age.ttf</fontPath>
```

forceCPURenderModeForDevices

Adobe AIR 3.7 och senare, endast iOS (valfritt)

Framtvinga CPU-återgivningläge för en angiven uppsättning enheter. Den här funktionen gör att du effektivt kan välja GPU-återgivningsläget för de återstående iOS-enheterna.

Du lägger till denna tagg som underordnad till `iPhone` -taggen och skapar en blankstegsavgränsad lista över enhetsmodellnamnen. Giltiga enhetsmodellnamn innehåller följande:

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		

Överordnat element:”iPhone” på sidan 226, ”initialWindow” på sidan 224

Underordnade element: inga

Innehåll
Blankstegsavgränsad lista över enhetsmodellnamnen.

Attribut:
Ingen.

Exempel
iOS:

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

fullScreen

Adobe AIR 2.0 och senare, iOS och Android – (valfritt)

Anger om programmet startas i helskärmsläge.

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll
`true` eller `false` (standard)

Exempel

```
<fullscreen>true</fullscreen>
```

height

Adobe AIR 1.0 och senare (valfritt)

Den inledande höjden på programmets huvudfönster.

Om du inte anger någon höjd hämtas den från inställningarna i SWF-rotfilen eller, om det är ett HTML-baserat AIR-program, från operativsystemet.

Den största tillåtna höjden för ett fönster har ändrats från 2 048 pixlar till 4 096 pixlar i AIR 2.

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll
Ett positivt heltal med ett största tillåtet värde på 4 095.

Exempel
```
<height>4095</height>
```

icon

Adobe AIR 1.0 och senare (valfritt)

`icon`-egenskapen anger en eller flera ikonfiler som ska användas för programmet. Du väljer själv om du vill inkludera ikoner. Om du inte anger någon `icon`-egenskap visar operativsystemet en standardikon.

Den angivna sökvägen är relativ till programmets rotkatalog. Ikonfilerna måste vara i PNG-format. Du kan ange följande ikonstorlekar:

Om det finns ett element för en viss storlek måste bilden i filen ha exakt den storleken som är angiven. Om inte alla storlekar är angivna anpassar operativsystemet bilden till den närmaste storleken så att den passar ikonens användning.

Obs! *De angivna ikonerna läggs inte automatiskt till i AIR-paketet. Ikonfilerna måste inkluderas på de motsvarande relativa platserna när programmet paketeras.*

För att få bästa resultat bör du ha en bild för varje tillgänglig storlek. Se dessutom till att ikonerna ser bra ut både i 16- och 32-bitars färglägen.

Överordnat element:”application” på sidan 205

Underordnade element:”imageNxN” på sidan 222

Innehåll
Ett imageNxN-element för varje önskad ikonstorlek.

Exempel
```
<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 och senare (krävs)

En identifieringssträng för programmet, som kallas program-ID. Ofta används en identifierare med omvänd DNS-notation, men det är inget krav.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll
ID-värdet är begränsat till följande tecken:

- 0–9
- a–z
- A–Z
- . (punkt)
- - (bindestreck)

Värdet måste innehålla 1 till 212 tecken. Det här elementet är obligatoriskt.

Exempel

```
<id>org.example.application</id>
```

imageNxN

Adobe AIR 1.0 och senare (valfritt)

Anger sökvägen till en ikon i förhållande till programkatalogen.

Följande ikonbilder kan användas, som alla anger olika ikonstorlekar:

- 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+)

Ikonen måste vara en PNG-bild som har exakt den storlek som bildelementet anger. Ikonfilerna måste ingå i programpaketet. Ikoner som programbeskrivningsfilen hänvisar till inkluderas inte automatiskt.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll

Sökvägen till ikonen kan innehålla alla Unicode-tecken (UTF-8) utom följande (eftersom de inte får användas som filnamn i olika filsystem):

Tecken	Hexadecimal kod
olika	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

Exempel

```
<image32x32>icons/icon32.png</image32x32>
```

InfoAdditions

Adobe AIR 1.0 och senare (valfritt)

Används för att ange ytterligare egenskaper för ett iOS-program.

Överordnat element:”iPhone” på sidan 226

Underordnade element: iOS-elementen Info.plist

Innehåll

Innehåller underordnade element som anger nyckelvärdepar som används som inställningar för Info.plist för programmet. Innehåll i InfoAdditions-elementet bör omslutas av ett CDATA-block.

Läs Information Property List Key Reference i Apples iPhone Reference Library om du vill ha mer information om nyckelvärdepar och hur du uttrycker dem i XML.

Exempel

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

Fler hjälpavsnitt

”Inställningar för iOS” på sidan 79

initialWindow

Adobe AIR 1.0 och senare (krävs)

Definierar den huvudsakliga innehållsfilen och det inledande programutseendet.

Överordnat element:”application” på sidan 205

Underordnade element: Alla nedanstående element kan visas som underordnade element för initialWindow-elementet. Vissa element kan dock ignoreras beroende på om AIR har stöd för fönster på plattformen:

Element	Skrivbord	Mobil
”aspectRatio” på sidan 208	ignoreras	används
”autoOrients” på sidan 209	ignoreras	används
”content” på sidan 210	används	används
”depthAndStencil” på sidan 212	används	används
”fullScreen” på sidan 220	ignoreras	används
”height” på sidan 220	används	ignoreras
”maximizable” på sidan 228	används	ignoreras
”maxSize” på sidan 228	används	ignoreras
”minimizable” på sidan 229	används	ignoreras
”minSize” på sidan 229	används	ignoreras
”renderMode” på sidan 231	används (AIR 3.0 och senare)	används
”requestedDisplayResolution” på sidan 232	används (AIR 3.6 och senare)	ignoreras
”resizable” på sidan 233	används	ignoreras
”softKeyboardBehavior” på sidan 233	ignoreras	används
”systemChrome” på sidan 235	används	ignoreras
”title” på sidan 236	används	ignoreras
”transparent” på sidan 237	används	ignoreras
”visible” på sidan 238	används	ignoreras
”width” på sidan 239	används	ignoreras
”x” på sidan 239	används	ignoreras
”y” på sidan 239	används	ignoreras

Innehåll

Underordnade element som definierar programutseende och -beteende.

Exempel

```
<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 och senare (valfritt)

Identifierar underkatalogen till standardinstallationskatalogen.

I Windows är standardinstallationens underkatalog mappen Program (Program Files). På Mac OS är det katalogen /Applications. I Linux är det /opt/. Om `installFolder`-egenskapen exempelvis har värdet `"Acme"` och programmets namn är `"ExampleApp"` installeras programmet i C:\Program Files\Acme\ExampleApp i Windows, i /Applications/Acme/Example.app i MacOS och i /opt/Acme/ExampleApp i Linux.

`installFolder`-egenskapen är valfri. Om du inte anger någon `installFolder`-egenskap installeras programmet i en underkatalog i standardinstallationskatalogen utifrån `name`-egenskapen.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll

Egenskapen `installFolder` kan innehålla alla Unicode-tecken (UTF-8), utom de som inte får användas som mappnamn i olika filsystem (en lista med undantag finns i avsnittet om `filename`-egenskapen).

Använd ett snedstreck (/) som katalogavgränsare om du vill ange en inbäddad underkatalog.

Exempel

```
<installFolder>utilities/toolA</installFolder>
```

iPhone

Adobe AIR 2.0, endast iOS – valfritt

Definierar iOS-specifika programegenskaper.

Överordnat element:”application” på sidan 205

Underordnade element:

- ”Entitlements” på sidan 214
- ”externalSwfs” på sidan 216
- ”forceCPURenderModeForDevices” på sidan 219
- ”InfoAdditions” på sidan 223
- ”requestedDisplayResolution” på sidan 232

Fler hjälpavsnitt

”Inställningar för iOS” på sidan 79

manifest

Adobe AIR 2.5 och senare, endast Android (valfritt)

Anger information som ska läggas till i Androids manifestfil för programmet.

Överordnat element:”manifestAdditions” på sidan 227

Underordnade element: Definieras av Android SDK.

Innehåll

Elementet manifest är inte, rent tekniskt, en del av AIR-programbeskrivningsschemat. Det är roten i Androids XML-manifestfil. Allt innehåll som du placerar i elementet manifest måste följa schemat för AndroidManifest.xml. När du genererar en APK-fil med AIR-verktygen kopieras informationen i manifest-elementet till motsvarande del av programmets genererade AndroidManifest.xml.

Om du anger Android-manifestvärden som bara är tillgängliga i en SDK-version som är senare än den som stöds direkt av AIR, måste du ange `-platformSDK`-flaggan som ADT när du paketerar programmet. Ange flaggan till filsystemsökvägen till en version av Android SDK som har stöd för de värden du lägger till.

Själva manifest-elementet måste omges av ett CDATA-block inuti AIR-programbeskrivningen.

Exempel

```
<![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>
]]>
```

Fler hjälpavsnitt

”Android-inställningar” på sidan 73

The AndroidManifest.xml File (Filen AndroidManifest.xml)

manifestAdditions

Adobe AIR 2.5 och senare, endast Android

Anger information som ska läggas till i Androids manifestfil.

Alla Android-program har en manifestfil, som definierar grundläggande programegenskaper. Androids manifestfil liknar begreppsmässigt AIR-programbeskrivningen. Ett AIR for Android-program har både en programbeskrivning och en automatiskt genererad manifestfil för Android. När ett AIR for Android-program paketeras läggs informationen i det här `manifestAdditions`-elementet till i motsvarande delar av Androids manifestfil.

Överordnat element:”android” på sidan 205

Underordnade element:”manifest” på sidan 226

Innehåll

Information i `manifestAdditions`-elementet läggs till i Androids XML-manifestfil.

AIR anger flera manifest-poster i den genererade manifestfilen för Android för att garantera att programmet och körningsfunktionerna fungerar korrekt. Du kan inte åsidosätta följande inställningar:

Du kan inte ange följande attribut för manifest-elementet:

- package
- android:versionCode
- android:versionName

Du kan inte ange följande attribut för det huvudsakliga activity-elementet:

- android:label
- android:icon

Du kan inte ange följande attribut för application-elementet:

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

Exempel

```
<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>
```

Fler hjälpavsnitt

”Android-inställningar” på sidan 73

The AndroidManifest.xml File (Filen AndroidManifest.xml)

maximizable

Adobe AIR 1.0 och senare (valfritt)

Anger om fönstret kan maximeras.

Obs! *På operativsystem som Mac OS X, där fönstermaximering är en storleksändring, måste både maximizable och resizable anges till* `false` *för att förhindra att fönstret kan zoomas eller storleksändras.*

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll

`true` (standard) eller `false`

Exempel

```
<maximizable>false</maximizable>
```

maxSize

Adobe AIR 1.0 och senare (valfritt)

Fönstrets största tillåtna storlekar. Om du inte anger någon maximal storlek fastställs den av operativsystemet.

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll

Två heltal som representerar den största bredden och höjden, åtskilda med mellanslag.

Obs! *Den maximala fönsterstorlek som stöds i AIR har ökat från 2 048 x 2 048 pixlar till 4 096 x 4 096 pixlar i AIR 2. (Eftersom skärmkoordinaterna är nollbaserade är det största värde du kan använda för bredd eller höjd 4 095.)*

Exempel

```
<maxSize>1024 360</maxSize>
```

minimizable

Adobe AIR 1.0 och senare (valfritt)

Anger om fönstret kan minimeras.

Överordnat element:"initialWindow" på sidan 224

Underordnade element: inga

Innehåll

`true` (standard) eller `false`

Exempel

```
<minimizable>false</minimizable>
```

minSize

Adobe AIR 1.0 och senare (valfritt)

Anger den minsta tillåtna storleken för fönstret.

Överordnat element:"initialWindow" på sidan 224

Underordnade element: inga

Innehåll

Två heltal som representerar den minsta bredden och höjden, åtskilda med mellanslag. Tänk på att den minsta storlek som tillåts av operativsystemet har högre prioritet än det värde som anges i programbeskrivningen.

Exempel

```
<minSize>120 60</minSize>
```

name

Adobe AIR 1.0 och senare (valfritt)

Det programnamn som visas i AIR-programmets installationsprogram.

Om inget `name`-element har angetts visar AIR-programmets installationsprogram `filename` som programnamn.

Överordnat element:"application" på sidan 205

Underordnade element:"text" på sidan 236

Innehåll

Om du anger en enda textnod (i stället för flera `<text>`-element) använder AIR-programmets installationsprogram det här namnet, oavsett systemspråk.

Programbeskrivningsschemat för AIR 1.0 tillåter bara att en enda textnod anges för namnet (inte flera `text`-element). I AIR 1.1 (eller senare) kan du ange flera språk i `name`-elementet.

`xml:lang`-attributet för varje text-element anger en språkkod enligt definitionen i RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

AIR-programmets installationsprogram använder det namn som mest liknar språket i användargränssnittet i användarens operativsystem. Ta till exempel en installation där `name`-elementet i programbeskrivningsfilen innehåller ett värde för språkinställningen en (engelska). AIR-programmets installationsprogram använder namnet en om operativsystemet identifierar en (engelska) som språk i användargränssnittet. Det använder också namnet en om språket i användargränssnittet är en-US (amerikansk engelska). Om språket i användargränssnittet är en-US, och programbeskrivningsfilen definierar både namnen en-US och en-GB, använder AIR-programmets installationsprogram värdet en-US. Om programmet inte definierar något namn som överensstämmer med språken i användargränssnittet, använder AIR-programmets installationsprogram det första `name`-värdet som är definierat i programbeskrivningsfilen.

`name`-elementet definierar bara den programtitel som används i AIR-programmets installationsprogram. AIR-programmets installationsprogram har stöd för flera olika språk: Traditionell kinesiska, förenklad kinesiska, tjeckiska, nederländska, engelska, franska, tyska, italienska, japanska, koreanska, polska, brasiliansk portugisiska, ryska, spanska, svenska och turkiska. AIR-programmets installationsprogram väljer visningsspråk (för annan text än programtitel och beskrivning) utifrån språket i användargränssnittet. Det här språkvalet är oberoende av inställningarna i programbeskrivningsfilen.

`name`-elementet definierar *inte* de språkinställningar som är tillgängliga för det installerade programmet som körs. Mer information om utveckling av flerspråkiga program finns i ”Lokalisera AIR-program” på sidan 288.

Exempel
I följande exempel definieras ett namn med en enda textnod:

```
<name>Test Application</name>
```

I följande exempel, som är giltigt i AIR 1.1 och senare, anges namnet på tre språk (engelska, franska och spanska) med <text>-elementnoder:

```
<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 och senare (krävs)

Identifierar namnet på en filtyp.

Överordnat element:”fileType” på sidan 217

Underordnade element: inga

Innehåll
En sträng som representerar namnet på filtypen.

Exempel

```
<name>adobe.VideoFile</name>
```

programMenuFolder

Adobe AIR 1.0 och senare (valfritt)

Anger var genvägar till programmet ska placeras på menyn Alla program i operativsystemet Windows eller på menyn Program i Linux. (Den här inställningen ignoreras för tillfället av andra operativsystem.)

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll

Den sträng som används för värdet `programMenuFolder` kan innehålla alla Unicode-tecken (UTF-8), utom de som inte får användas som mappnamn i olika filsystem (en lista med undantag finns i avsnittet om `filename`-elementet). Använd *inte* ett snedstreck (/) som sista tecken i det här värdet.

Exempel

```
<programMenuFolder>Example Company/Sample Application</programMenuFolder>
```

publisherID

Adobe AIR 1.5.3 och senare (valfritt)

Identifierar utgivar-ID:t för uppdatering av ett AIR-program som ursprungligen skapats med AIR 1.5.2 eller en tidigare version.

Ange bara utgivar-ID när du skapar en programuppdatering. Värdet på elementet `publisherID` måste matcha det utgivar-ID som genererats av AIR för den tidigare versionen av programmet. För ett installerat program finns utgivar-ID:t i den mapp i vilken programmet installeras, i filen `META-INF/AIR/publisherid`.

Nya program som skapas med AIR 1.5.3 eller senare bör inte ange något utgivar-ID.

Mer information finns i ”Om utgivaridentifierare i AIR” på sidan 187.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll

En sträng med utgivar-ID.

Exempel

```
<publisherID>B146A943FBD637B68C334022D304CEA226D129B4.1</publisherID>
```

renderMode

Adobe AIR 2.0 och senare (valfritt)

Anger om GPU-acceleration ska användas när det stöds på den aktuella enheten.

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll
Ett av följande värden:

- `auto` (standard) – för tillfället används CPU-läge som reservalternativ.
- `cpu` – maskinvaruacceleration används inte.
- `direct` – återgivningsdisposition inträffar i CPU; blittning används i GPU. Tillgängligt i AIR 3+.

  Obs! *För att kunna anpassa GPU-accelerationen för Flash-innehållet med AIR för mobila plattformar, rekommenderar Adobe att du använder renderMode="direct" (dvs. Stage3D) i stället för renderMode="gpu". Adobe stöder och rekommenderar följande Stage3D-baserade ramverk officiellt: Starling (2D) och Away3D (3D). Mer information om Stage3D och Starling/Away3D finns på http://gaming.adobe.com/getstarted/.*
- `gpu` – maskinvaruacceleration används om det är tillgängligt.

  Viktigt! *Använd inte GPU-återgivningsläget för Flex-program.*

Exempel

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

requestedDisplayResolution

Adobe AIR 2.6 och senare versioner, endast iOS; Adobe AIR 3.6 och senare versioner, OS X – Valfritt

Anger om programmet ska använda standardupplösning eller hög upplösning på en enhet eller dator med högupplöst skärm. Om värdet anges som *standard* (standard) visas skärmen som en skärm med standardupplösning för programmet. Om värdet anges som *high* kan programmet adressera varje högupplöst pixel.

Om inställningen på en högupplöst 640 x 960 iPhone-skärm till exempel är *standard* är scenens helskärmsmått 320 x 480, och varje programpixel återges med fyra skärmpixlar. Om inställningen är *high* är scenens helskärmsmått 640 x 960.

På enheter med skärmar med standardupplösning matchar scenmåtten skärmens mått, oavsett vilken inställning som används.

Om elementet `requestedDisplayResolution` har kapslats i elementet `iPhone` gäller det iOS-enheter. I så fall kan attributet `excludeDevices` användas för att ange enheter för vilka inställningen inte ska användas.

Om elementet `requestedDisplayResolution` har kapslats i elementet `initialWindow` gäller det AIR-skrivbordsprogram på MacBook Pro-datorer med stöd för en högupplöst skärm. Det angivna värdet gäller alla systemspecifika fönster som används i programmet. Kapsling av elementet `requestedDisplayResolution` i elementet `initialWindow` stöds i AIR 3.6 och senare versioner.

Överordnat element:”iPhone” på sidan 226, ”initialWindow” på sidan 224

Underordnade element: inga

Innehåll
Antingen *standard* (standardvärdet) eller *high*.

Attribut:
excludeDevices – en blankstegsavgränsad lista över iOS-modellnamn eller modellnamnsprefix. Detta gör det möjligt för utvecklare att låta vissa enheter använda hög upplösning medan andra använder standardupplösning. Det här attributet är bara tillgängligt på iOS (elementet `requestedDisplayResolution` kapslas i elementet `iPhone`). Attributet `excludeDevices` finns i AIR 3.6 och senare versioner.

För alla enheter vars modellnamn anges i det här attributet är värdet för `requestedDisplayResolution` motsatsen till det angivna värdet. Om värdet `requestedDisplayResolution` är *high* använder alltså de uteslutna enheterna (excludedDevices) standardupplösning. Om värdet `requestedDisplayResolution` är *standard* använder de uteslutna enheterna hög upplösning.

Värdena är iOS-enheternas modellnamn eller modellnamnsprefix. Värdet iPad3,1 avser till exempel just en 3:e generationens iPad med Wi-Fi (men inte en 3:e generationens iPad med GSM eller CDMA). Värdet iPad3 avser alla 3:e generationens iPad-enheter. En inofficiell lista över iOS-modellnamn finns på iPhone wiki-sidan med modeller.

Exempel
Skrivbord:

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

iOS:

```
<iPhone>
    <requestedDisplayResolution excludeDevices="iPad3
iPad4">high</requestedDisplayResolution>
</iPhone>
```

resizable

Adobe AIR 1.0 och senare (valfritt)

Anger om det går att ändra storlek på fönstret.

Obs! *På operativsystem som Mac OS X, där fönstermaximering är en storleksändring, måste både maximizable och resizable anges till* `false` *för att förhindra att fönstret kan zoomas eller storleksändras.*

Överordnat element:”initialWindow” på sidan 224

Underordnade element:

Innehåll
`true` (standard) eller `false`

Exempel

```
<resizable>false</resizable>
```

softKeyboardBehavior

Adobe AIR 2.6 och senare, mobilprofil (valfritt)

Anger programmets standardbeteende när ett virtuellt tangentbord visas. Standardbeteendet är att panorera programmet uppåt. Miljön ser till att det textfält eller interaktiva objekt som har fokus finns kvar på skärmen. Använd alternativet *pan* om ditt program inte har egen logik för att hantera tangentbord.

Du kan också inaktivera det automatiska beteendet genom att ange elementet `softKeyboardBehavior` som *none*. I så fall skickar textfält och interaktiva objekt en SoftKeyboardEvent när skärmtangentbordet öppnas, men miljön varken panorerar eller ändrar storlek på programmet. Ditt program måste se till att textinmatningsområdet är synligt.

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll
Antingen *none* eller *pan*. Standardvärdet är *pan*.

Exempel
```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Fler hjälpavsnitt
SoftKeyboardEvent

supportedLanguages
Adobe AIR 3.2 och senare (valfritt)

Anger det språk som programmet har stöd för. Det här elementet används bara av iOS, låsta Mac-miljöer och Android-program. Elementet ignoreras av alla andra programtyper.

Om du inte anger det här elementet utförs som standard följande åtgärder baserat på programtyp vid paketering:

- iOS – Alla språk som stöds av AIR-miljön listas i iOS App Store som språk som stöds för programmet.
- Låst Mac-miljö – Program som paketeras med ett låst miljöpaket har ingen lokaliseringsinformation.
- Android – Programpaketet har resurser för alla språk som stöds av AIR-miljön.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll
En blankstegsavgränsad lista över språk som stöds. Giltiga språkvärden är ISO 639-1-värden för de språk som stöds av AIR-miljön: `en`, `de`, `es`, `fr`, `it`, `ja`, `ko`, `pt`, `ru`, `cs`, `nl`, `pl`, `sv`, `tr`, `zh`, `da`, `nb`, `iw`.

Paketeraren genererar ett felmeddelande om värdet för `<supportedLanguages>`-elementet är tomt.

Obs! *Lokaliserade taggar (t.ex. name-taggen) ignorerar värdet på ett språk om du använder taggen* `<supportedLanguages>` *och den inte innehåller det språket. Om ett systemspecifikt tillägg har resurser för ett språk som inte anges av taggen* `<supportedLangauges>` *visas ett varningsmeddelande och resurserna ignoreras för det språket.*

Exempel
```
<supportedLanguages>en ja fr es</supportedLanguages>
```

supportedProfiles
Adobe AIR 2.0 och senare (valfritt)

Identifierar de profiler som stöds i programmet.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll

Följande värden kan inkluderas i `supportedProfiles`-elementet:

- `desktop` – Skrivbordsprofilen används för AIR-program som installeras på en stationär dator med hjälp av en AIR-fil. Dessa program har inte tillgång till klassen NativeProcess (som används för att kommunicera med systemspecifika program).
- `extendedDesktop` – Den utökade skrivbordsprofilen används för AIR-program som installeras på en stationär dator med hjälp av ett systemspecifikt installationsprogram. Dessa program har tillgång till klassen NativeProcess (som används för att kommunicera med systemspecifika program).
- `mobileDevice` – Mobilprofilen används för mobilprogram.
- `extendedMobileDevice` – Den utökande profilen för mobila enheter används inte för närvarande.

`SupportedProfiles`-egenskapen är valfri. När du inte tar med detta element i programbeskrivningsfilen, kan programmet kompileras och distribueras för alla profiler.

Om du vill ange flera profiler ska du separera dem med ett blanktecken. I följande inställning visas exempelvis att programmet endast är tillgängligt för desktop-profiler och utökade profiler:

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

Obs! *När du kör ett program med ADL och inte anger något värde för ADL-alternativet* `-profile` *används den första profilen i programbeskrivningen. (Om inga profiler anges i programbeskrivningen heller används skrivbordsprofilen.)*

Exempel

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

Fler hjälpavsnitt

”Enhetsprofiler” på sidan 241

”Profiler som stöds” på sidan 72

systemChrome

Adobe AIR 1.0 och senare (valfritt)

Anger om det inledande programfönstret skapas med standardnamnlist, standardkanter och standardkontroller från operativsystemet.

Fönstrets systeminställningar kan inte ändras vid körning.

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll

Ett av följande värden:

- `none` – Inga systemkontroller har angetts. Programmet (eller ett programramverk som Flex) hanterar visningen av fönsterkontroller.
- `standard` (standardvärdet) – Systemkontroller anges av operativsystemet.

Exempel

```
<systemChrome>standard</systemChrome>
```

text

Adobe AIR 1.1 och senare (valfritt)

Anger en lokaliserad sträng.

Attributet `xml:lang` för ett textelement anger en språkkod, enligt definitionen i RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

AIR-programmets installationsprogram använder elementet `text` med det värde för `xml:lang`-attributet som mest liknar språket i användargränssnittet i användarens operativsystem.

Som exempel kan nämnas en installation där elementet `text` innehåller ett värde för språket en (engelska). AIR-programmets installationsprogram använder namnet en om operativsystemet identifierar en (engelska) som språk i användargränssnittet. Det använder också namnet en om språket i användargränssnittet är en-US (amerikansk engelska). Om språket i användargränssnittet är en-US, och programbeskrivningsfilen definierar både namnen en-US och en-GB, använder AIR-programmets installationsprogram värdet en-US.

Om programmet inte definierar något `text`-element som matchar språken i användargränssnittet, använder AIR-programmets installationsprogram det första `name`-värde som definierats i programbeskrivningsfilen.

Överordnade element:

- ”name” på sidan 229
- ”description” på sidan 213

Underordnade element: inga

Innehåll

Ett `xml:lang`-attribut som anger ett språk och en sträng med lokaliserad text.

Exempel

```
<text xml:lang="fr">Bonjour AIR</text>
```

title

Adobe AIR 1.0 och senare (valfritt)

Anger det namn som visas i namnlisten i det inledande programfönstret.

Det visas bara ett namn om elementet `systemChrome` har angetts som `standard`.

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll

En sträng som innehåller fönsternamnet.

Exempel

```
<title>Example Window Title</title>
```

transparent

Adobe AIR 1.0 och senare (valfritt)

Anger om det inledande programfönstret alfablandas med skrivbordet.

Ett fönster med genomskinlighet aktiverat kan ritas långsammare och kräva mer minne. Inställningen för genomskinlighet kan inte ändras under körning.

Viktigt! *Du kan bara ange* `transparent` *som* `true` *om* `systemChrome` *är* `none`*.*

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll

`true` eller `false` (standard)

Exempel

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

version

Adobe AIR 1.0 till 2.0 (krävs); inte tillåtet i AIR 2.5 och senare

Anger versionsinformation för programmet.

Versionssträngen är en programdefinierad beteckning. AIR tolkar inte versionssträngen på något sätt. Version ”3.0” antas därför inte vara mer aktuell än version ”2.0”. Exempel: `"1.0"`, `".4"`, `"0.5"`, `"4.9"`, `"1.3.4a"`.

I AIR 2.5 och senare versioner har elementet `version` ersatts av elementen `versionNumber` och `versionLabel`.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll

En sträng som innehåller programmets version.

Exempel

```
<version>0.1 Alpha</version>
```

versionLabel

Adobe AIR 2.5 och senare (valfritt)

Anger en läslig versionssträng.

Värdet på versionsetiketten visas i installationsdialogrutor i stället för värdet på elementet `versionNumber`. Om `versionLabel` inte används används `versionNumber` för båda.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll
En sträng som innehåller den allmänna versionstexten.

Exempel
```
<versionLabel>0.9 Beta</versionlabel>
```

versionNumber

Adobe AIR 2.5 och senare (krävs)

Programmets versionsnummer.

Överordnat element:”application” på sidan 205

Underordnade element: inga

Innehåll
Versionsnumret kan innehålla en sekvens med upp till tre heltal som skiljs åt med punkter. Varje heltal måste vara ett tal från 0 till och med 999.

Exempel
```
<versionNumber>1.0.657</versionNumber>

<versionNumber>10</versionNumber>

<versionNumber>0.01</versionNumber>
```

visible

Adobe AIR 1.0 och senare (valfritt)

Anger om det inledande programfönstret är synligt så fort det skapas.

AIR-fönster, även det inledande fönstret, skapas som standard med en osynlig status. Du kan visa ett fönster genom att anropa metoden `activate()` för NativeWindow-objektet eller genom att ange egenskapen `visible` som `true`. Det kan vara bra att låta huvudfönstret vara dolt i början så att eventuella ändringar i fönstrets position och storlek samt layouten för dess innehåll inte visas.

Flex-komponenten `mx:WindowedApplication` visar och aktiverar fönstret automatiskt direkt innan `applicationComplete`-händelsen skickas, om inte `visible`-attributet är angett som `false` i MXML-definitionen.

I mobilprofilen på enheter, där stöd för fönster saknas, ignoreras inställningen för visible.

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll
`true` eller `false` (standard)

Exempel
```
<visible>true</visible>
```

width

Adobe AIR 1.0 och senare (valfritt)

Den inledande bredden på programmets huvudfönster.

Om du inte anger någon bredd hämtas den från inställningarna i SWF-rotfilen eller, om det är ett HTML-baserat AIR-program, från operativsystemet.

Den största tillåtna bredden för ett fönster har ändrats från 2 048 pixlar till 4 096 pixlar i AIR 2.

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll
Ett positivt heltal med ett största tillåtet värde på 4 095.

Exempel

```
<width>1024</width>
```

x

Adobe AIR 1.0 och senare (valfritt)

Det inledande programfönstrets vågräta position.

Oftast är det bäst att låta operativsystemet avgöra fönstrets inledande position i stället för att ange ett fast värde.

Origo i skärmkoordinatsystemet (0,0) är det övre vänstra hörnet av skrivbordsskärmen (fastställs av operativsystemet).

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll
Ett heltalsvärde.

Exempel

```
<x>120</x>
```

y

Adobe AIR 1.0 och senare (valfritt)

Det inledande programfönstrets lodräta position.

Oftast är det bäst att låta operativsystemet avgöra fönstrets inledande position i stället för att ange ett fast värde.

Origo i skärmkoordinatsystemet (0,0) är det övre vänstra hörnet av skrivbordsskärmen (fastställs av operativsystemet).

Överordnat element:”initialWindow” på sidan 224

Underordnade element: inga

Innehåll

Ett heltalsvärde.

Exempel

```
<y>250</y>
```

Kapitel 15: Enhetsprofiler

Adobe AIR 2 och senare

Profiler är ett sätt att definiera de typer av enheter på vilka ditt program kan användas. En profil definierar en uppsättning API:er och funktioner som vanligtvis stöds på en viss typ av enhet. De tillgängliga profilerna är:

- desktop
- extendedDesktop
- mobileDevice
- extendedMobileDevice

Du kan definiera profiler för ditt program i programbeskrivningen. Användare av datorer och enheter i de inkluderade profilerna kan installera ditt program, vilket användare av andra enheter inte kan. Om du till exempel bara inkluderar skrivbordsprofilen i programbeskrivningen kan användare bara installera och köra ditt program på stationära datorer.

Om du inkluderar en profil som ditt program egentligen inte har fullt stöd för kan det innebära att användare i en sådan miljö får en sämre upplevelse. Om du inte anger någon profil alls i programbeskrivningen begränsas ditt program inte av AIR. Du kan paketera programmet i något av de format som stöds, och användare med enheter från valfri profil kan installera det, men det är inte säker att det fungerar korrekt vid körning.

När det är möjligt används profilbegränsningarna när du paketerar programmet. Om du till exempel bara inkluderar profilen extendedDesktop kan du inte paketera programmet som en AIR-fil, utan bara som ett systemspecifikt installationsprogram. Enligt samma princip kan du inte heller paketera programmet som en APK-fil för Android om du inte inkluderar profilen mobileDevice.

En och samma enhet kan ha stöd för fler än en profil. AIR på stationära datorer har till exempel stöd för program i både profilen desktop och profilen extendedDesktop. Men ett program i profilen extendedDesktop kan kommunicera med systemspecifika processer och MÅSTE paketeras som ett systemspecifikt installationsprogram (exe, dmg, deb eller rpm). Däremot kan ett program i skrivbordsprofilen (desktop) inte kommunicera med systemspecifika processer. Ett program i skrivbordsprofilen kan paketeras antingen som en AIR-fil eller som ett systemspecifikt installationsprogram.

Om en funktion ingår i en profil betyder det att stöd för den funktionen vanligen finns på den typ av enheter för vilka profilen är avsedd. Det betyder däremot inte att alla enheter i profilen har stöd för alla funktioner. De flesta, men inte alla, mobiltelefoner har till exempel en accelerometer. Klasser och funktioner som inte har universellt stöd har oftast en boolesk egenskap, som du kan kontrollera innan du använder funktionen. I fallet med accelerometern kan du till exempel testa den statiska egenskapen `Accelerometer.isSupported` för att avgöra om den aktuella enheten har en accelerometer som stöds.

Följande profiler kan tilldelas till ditt AIR-program med hjälp av elementet supportedProfiles i programbeskrivningen:

Skrivbord Skrivbordsprofilen definierar ett antal funktioner för AIR-program som är installerade som AIR-filer på en stationär dator. Dessa program installeras och körs på de skrivbordsplattformar som stöds (Mac OS, Windows och Linux). AIR-program som utvecklats i AIR-versioner före AIR 2 betraktas som om de tillhör skrivbordsprofilen. Vissa API:er fungerar inte i den här profilen. Skrivbordsprogram kan t.ex. inte kommunicera med interna processer.

Utökat skrivbord Den utökade skrivbordsprofilen definierar ett antal funktioner för AIR-program som paketeras i och installeras med en systemspecifik installationsfil. Dessa systemspecifika installationsfiler är EXE-filer i Windows, DMG-filer i Mac OS och BIN-, DEB- eller RPM-filer i Linux. Utökade skrivbordsprogram har extra funktioner som

inte finns i program med skrivbordsprofilen. Du hittar mer information i ”Paketera ett systemspecifikt installationsprogram för skrivbordet” på sidan 55.

Mobil enhet Profilen för mobila enheter definierar en uppsättning funktioner för program som är installerade på mobila enheter som exempelvis mobiltelefoner och surfplattor. Dessa program installeras och körs på kompatibla mobilplattformar såsom Android, Blackberry Tablet OS och iOS.

Utökad mobil enhet Den utökade mobilprofilen definierar ett antal utökade funktioner för program som installeras på mobila enheter. För närvarande finns det inga enheter som har stöd för den här profilen.

Begränsa målprofiler i programbeskrivningsfilen

Adobe AIR 2 och senare

Från och med AIR 2 inkluderar programbeskrivningsfilen elementet `supportedProfiles`, som du kan använda för att begränsa målprofiler. I följande inställning visas exempelvis att programmet endast är tillgängligt för skrivbordsprofiler:

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

När elementet är angett kan programmet bara paketeras i de profiler som du anger. Använd följande värden:

- `desktop` – skrivbordsprofilen
- `extendedDesktop` – den utökade skrivbordsprofilen
- `mobileDevice` – mobilprofilen

Elementet `supportedProfiles` är valfritt. När du inte tar med detta element i programbeskrivningsfilen, kan programmet paketeras och distribueras för alla profiler.

Om du vill ange flera profiler i elementet `supportedProfiles` kan du avgränsa varje profil med ett mellanslagstecken så som visas nedan:

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

Funktioner i de olika profilerna

Adobe AIR 2 och senare

I följande tabell visas de klasser och funktioner som inte stöds i alla profiler.

Klass eller funktion	desktop	extendedDesktop	mobileDevice
Accelerometer (Accelerometer.isSupported)	Nej	Nej	Kontrollera
Accessibility (Capabilities.hasAccessibility)	Ja	Ja	Nej
Akustisk ekoreducering (Microphone.getEnhancedMicrophone())	Ja	Ja	Nej
ActionScript 2	Ja	Ja	Nej
CacheAsBitmap-matris	Nej	Nej	Ja

Klass eller funktion	desktop	extendedDesktop	mobileDevice
Camera (Camera.isSupported)	Ja	Ja	Ja
CameraRoll	Nej	Nej	Ja
CameraUI (CameraUI.isSupported)	Nej	Nej	Ja
Låsta miljöpaket	Ja	Ja	Ja
ContextMenu (ContextMenu.isSupported)	Ja	Ja	Nej
DatagramSocket (DatagramSocket.isSupported)	Ja	Ja	Ja
DockIcon (NativeApplication.supportsDockIcon)	Kontrollera	Kontrollera	Nej
Drag-and-drop (NativeDragManager.isSupported)	Ja	Ja	Kontrollera
EncyptedLocalStore (EncyptedLocalStore.isSupported)	Ja	Ja	Ja
Flash Access (DRMManager.isSupported)	Ja	Ja	Nej
GameInput (GameInput.isSupported)	Nej	Nej	Nej
Geolocation (Geolocation.isSupported)	Nej	Nej	Kontrollera
HTMLLoader (HTMLLoader.isSupported)	Ja	Ja	Nej
IME (IME.isSupported)	Ja	Ja	Kontrollera
LocalConnection (LocalConnection.isSupported)	Ja	Ja	Nej
Microphone (Microphone.isSupported)	Ja	Ja	Kontrollera
Flerkanalsljud (Capabilities.hasMultiChannelAudio())	Nej	Nej	Nej
ANE-tillägg	Nej	Ja	Ja
NativeMenu (NativeMenu.isSupported)	Ja	Ja	Nej
NativeProcess (NativeProcess.isSupported)	Nej	Ja	Nej
NativeWindow (NativeWindow.isSupported)	Ja	Ja	Nej
NetworkInfo (NetworkInfo.isSupported)	Ja	Ja	Kontrollera
Öppna filer med standardprogram	Begränsat	Ja	Nej
PrintJob (PrintJob.isSupported)	Ja	Ja	Nej
SecureSocket (SecureSocket.isSupported)	Ja	Ja	Kontrollera
ServerSocket (ServerSocket.isSupported)	Ja	Ja	Ja
Shader	Ja	Ja	Begränsat
Stage3D (Stage.stage3Ds.length)	Ja	Ja	Ja
Scenorientering (Stage.supportsOrientationChange)	Nej	Nej	Ja
StageVideo	Nej	Nej	Kontrollera

Klass eller funktion	desktop	extendedDesktop	mobileDevice
StageWebView (StageWebView.isSupported)	Ja	Ja	Ja
Starta program vid inloggning (NativeApplication.supportsStartAtLogin)	Ja	Ja	Nej
StorageVolumeInfo (StorageVolumeInfo.isSupported)	Ja	Ja	Nej
Systemets vänteläge	Nej	Nej	Ja
SystemTrayIcon (NativeApplication.supportsSystemTrayIcon)	Kontrollera	Kontrollera	Nej
TLF-indata (Text Layout Framework)	Ja	Ja	Nej
Updater (Updater.isSupported)	Ja	Nej	Nej
XMLSignatureValidator (XMLSignatureValidator.isSupported)	Ja	Ja	Nej

Förklaring till posterna i tabellen:

- *Kontrollera* – Funktionen stöds på vissa, men inte alla, enheter i profilen. Du bör kontrollera vid körning om funktionen stöds innan du använder den.
- *Begränsat* – Funktionen stöds, men har avsevärda begränsningar. Läs relevant dokumentation för mer information.
- *Nej* – Funktionen stöds inte i profilen.
- *Ja* – Funktionen stöds i profilen. Tänk på att vissa enskilda enheter kan sakna den maskinvara som krävs för en viss funktion. Alla telefoner har till exempel inte en kamera.

Ange profiler för felsökning med ADL

Adobe AIR 2 och senare

ADL kontrollerar om du anger profiler som stöds i elementet `supportedProfiles` i programbeskrivningsfilen. Om du gör det, använder ADL som standard den första profilen som anges vid felsökning.

Du kan ange en profil för ADL felsökningssessionen med kommandoradsargumentet `-profile`. (Läs mer i ”AIR Debug Launcher (ADL)” på sidan 156.) Du kan använda argumentet oavsett om du vill ange en profil i elementet `supportedProfiles` i programbeskrivningsfilen eller inte. Om du anger elementet `supportedProfiles` måste det dock inkludera profilen som du anger på kommandoraden. I annat fall kommer ett fel att genereras i ADL.

Kapitel 16: Webbläsar-API:er i AIR.SWF

Anpassa badge.swf för sömlös installation

Förutom att använda filen badge.swf som ingår i SDK kan du skapa en egen SWF-fil och använda den i en webbläsare. Den egna SWF-filen kan samverka med körningsversionen på följande sätt:

- Den kan installera ett AIR-program. Se ”Installera ett AIR-program från webbläsaren” på sidan 250.
- Den kan kontrollera om ett visst AIR-program är installerat. Se ”Kontrollera från en webbsida om ett AIR-program är installerat” på sidan 249.
- Den kan kontrollera om körningsversionen är installerad. Se ”Kontrollera om körningsversionen är installerad” på sidan 249.
- Den kan starta ett installerat AIR-program i användarens system. Se ”Starta ett installerat AIR-program från webbläsaren” på sidan 251.

Du får tillgång till dessa funktioner genom att anropa API:er i en SWF-fil som finns på adobe.com: air.swf. Du kan anpassa filen badge.swf och anropa API:er för air.swf från din egen SWF-fil.

En SWF-fil som körs i webbläsaren kan dessutom kommunicera med ett AIR-program som körs genom att använda klassen LocalConnection. Mer information finns i Kommunicera med andra Flash Player- och AIR-instanser (för ActionScript-utvecklare) eller Communicating with other Flash Player and AIR instances (för HTML-utvecklare).

Viktigt! *Funktionerna som beskrivs i det här avsnittet (och API:erna i filen air.swf) kräver att Adobe® Flash® Player 9 uppdatering 3 eller senare är installerat i slutanvändarens webbläsare på Windows eller Mac OS. I Linux krävs Flash Player 10 (version 10.0.12.36 eller senare) för sömlös installation. Du kan skriva kod om du vill kontrollera den installerade versionen av Flash Player och skapa ett alternativt gränssnitt till användaren om den version av Flash Player som krävs inte är installerad. Om till exempel en äldre version av Flash Player är installerad, kan du skapa en länk till hämtningsversionen av AIR-filen (i stället för att använda filen badge.swf eller API:n för air.swf för att installera ett program).*

Installera ett AIR-program med filen badge.swf

I AIR SDK och Flex SDK finns filen badge.swf som gör det enklare att använda funktionen för sömlös installation. Filen badge.swf kan installera körningsversionen och ett AIR-program från en länk på en webbsida. Du kan distribuera filen badge.swf och dess källkod på din webbplats.

Bädda in filen badge.swf på en webbsida

1. Leta reda på följande filer, som finns i katalogen samples/badge i AIR SDK eller Flex SDK, och lägg till dem på webbservern.
   - badge.swf
   - default_badge.html
   - AC_RunActiveContent.js
2. Öppna sidan default_badge.html i en textredigerare.

3 Gå till sidan default_badge.html och JavaScript-funktionen `AC_FL_RunContent()`, och justera `FlashVars`-parameterdefinitionerna för följande:

Parameter	Beskrivning
`appname`	Namnet på programmet, som visas av SWF-filen när körningsversionen inte är installerad.
`appurl`	(Obligatoriskt.) URL:en till AIR-filen som hämtas. Du måste använda en absolut URL, inte en relativ.
`airversion`	(Obligatoriskt.) Ställ in detta på 1.0 för version 1.0 av körningsversionen.
`imageurl`	URL:en till bilden som ska visas i emblemet (valfritt).
`buttoncolor`	Färgen på hämtningsknappen (anges som ett hexadecimalt värde, till exempel `FFCC00`).
`messagecolor`	Färgen på textmeddelandet som ska visas under knappen när körningsversionen inte är installerad (anges som ett hexadecimalt värde, till exempel `FFCC00`).

4 Filen badge.swf måste vara minst 217 pixlar bred och 180 pixlar hög. Justera värdena för parametrarna `width` och `height` för funktionen `AC_FL_RunContent()` efter behov.

5 Byt namn på filen default_badge.html och justera koden (eller infoga den på en annan HTML-sida) efter eget behov.

Obs! *För HTML-taggen `embed`, som används för att läsa in filen badge.swf, ska du inte ställa in attributet `wmode`, utan låta den behålla standardinställningen (`"window"`). Andra `wmode`-inställningar kommer att förhindra installationen i vissa system. Dessutom kommer andra `wmode`-inställningar att generera ett felmeddelande: "Error #2044: Unhandled ErrorEvent:. text=Error #2074: The stage is too small to fit the download ui."*

Du kan också redigera och kompilera om filen badge.swf. Mer information finns i ”Redigera filen badge.swf” på sidan 247.

Installera AIR-programmet från en sömlös installationslänk på en webbsida

När du har lagt till den sömlösa installationslänken på en sida, kan användaren installera AIR-programmet genom att klicka på länken i SWF-filen.

1 Gå till HTML-sidan i en webbläsare som har Flash Player (version 9 uppdatering 3 eller senare i Windows och Mac OS, eller version 10 i Linux) installerat.

2 Klicka på länken i filen badge.swf på webbsidan.

- Om du har installerat körningsversionen går du vidare till nästa steg.
- Om du inte har installerat körningsversionen visas en dialogruta med en fråga om du vill installera den. Installera körningsversionen (se ”Installation av Adobe AIR” på sidan 3) och fortsätt med nästa steg.

3 Låt standardinställningarna vara markerade i installationsfönstret och klicka sedan på Fortsätt.

På en Windows-dator gör AIR följande automatiskt:

- Installerar programmet i c:\Program\
- Skapar en genväg till programmet på skrivbordet
- Skapar en genväg på Start-menyn
- Lägger till en post för programmet i Lägg till/ta bort program på Kontrollpanelen

I Mac OS lägger installationsprogrammet till programmet i katalogen Applications (till exempel i katalogen /Applications i Mac OS).

På en Linux-dator gör AIR följande automatiskt:

- Installerar programmet i /opt.
- Skapar en genväg till programmet på skrivbordet
- Skapar en genväg på Start-menyn
- Lägger till en post för programmet i systemets pakethanterare.

4 Välj önskade alternativ och klicka på Installera.

5 Klicka på Slutför när installationen är klar.

Redigera filen badge.swf

Flex SDK och AIR SDK innehåller källfilerna för filen badge.swf. Dessa filer finns i mappen samples/badge i SDK-paketet:

Källfiler	Beskrivning
badge.fla	Flash -källfilen som används för att kompilera filen badge.swf. Filen badge.fla kompileras till en SWF 9-fil (som kan läsas in i Flash Player).
AIRBadge.as	En ActionScript 3.0-klass som definierar basklassen som används i filen basdge.fla.

Du kan använda Flash Professional om du vill designa om det visuella gränssnittet för filen badge.fla.

Konstruktorfunktionen `AIRBadge()`, som definieras i klassen AIRBadge, läser in filen air.swf som finns på http://airdownload.adobe.com/air/browserapi/air.swf. Filen air.swf innehåller kod för funktionen för sömlös installation.

Metoden `onInit()` (i klassen AIRBadge) anropas när filen air.swf har lästs in:

```
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;
    }
}
```

Koden ställer in den globala `_air`-variabeln på huvudklassen för den inlästa air.swf-filen. Den här klassen omfattar följande offentliga metoder, som badge.swf använder för att anropa funktionen för sömlös installation

Metod	Beskrivning
`getStatus()`	Tar reda på om körningsversionen är installerad (eller kan installeras) på datorn. Mer information finns i "Kontrollera om körningsversionen är installerad" på sidan 249. • `runtimeVersion` – En sträng som anger vilken version av miljön (till exempel `"1.0.M6"`) som krävs för det program som ska installeras.
`installApplication()`	Installerar angivet program på användarens dator. Mer information finns i "Installera ett AIR-program från webbläsaren" på sidan 250. • `url` – En sträng som definierar URL:en. Du måste använda en absolut URL-sökväg, inte en relativ. • `runtimeVersion` – En sträng som anger vilken version av miljön (till exempel `"2.5"`) som krävs för det program som ska installeras. • `arguments` – Argument som ska skickas till programmet om det startar vid installationen. Programmet startar vid installationen om elementet `allowBrowserInvocation` ställs in på `true` i programbeskrivningsfilen. (Mer information om programbeskrivningsfilen finns i "AIR-programbeskrivningsfiler" på sidan 200.) Om programmet startar med anledning av en sömlös installation från webbläsaren (där användaren väljer att starta vid installation), skickar programmets NativeApplication-objekt ett BrowserInvokeEvent-objekt bara om argument skickas. Ta hänsyn till datasäkerheten när du skickar programmet. Mer information finns i "Starta ett installerat AIR-program från webbläsaren" på sidan 251.

Inställningarna för `url` och `runtimeVersion` skickas till SWF-filen via FlashVars-inställningarna i HTML-sidan som fungerar som behållare.

Om programmet startar automatiskt vid installationen kan du använda LocalConnection-kommunikation så att det installerade programmet kontaktar filen badge.swf vid ett anrop. Mer information finns i Kommunicera med andra Flash Player- och AIR-instanser (för ActionScript-utvecklare) eller Communicating with other Flash Player and AIR instances (för HTML-utvecklare).

Du kan också anropa metoden `getApplicationVersion()` för filen air.swf om du vill kontrollera om ett program är installerat. Du kan anropa den här metoden innan programmet installeras eller efter att installationen har startat. Mer information finns i "Kontrollera från en webbsida om ett AIR-program är installerat" på sidan 249.

Läsa in filen air.swf

Du kan skapa en egen SWF-fil som använder API:erna i filen air.swf om du vill samverka med körnings- och AIR-programmen från en webbsida i en webbläsare. Filen air.swf finns på http://airdownload.adobe.com/air/browserapi/air.swf. Om du vill referera till API:erna för air.swf från SWF-filen läser du in air.swf till samma programdomän som SWF-filen. I följande kod visas ett exempel på hur filen air.swf läses in i programdomänen när SWF-filen läses in:

```
 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;
}
```

När filen air.swf har lästs in (när Loader-objektets `contentLoaderInfo`-objekt skickar händelsen `init`), kan du anropa någon av API:erna för air.swf, vilket beskrivs i nedanstående avsnitt.

Obs! *Filen badge.swf, som finns i AIR SDK och i Flex SDK, läser automatiskt in filen air.swf. Se "Installera ett AIR-program med filen badge.swf" på sidan 245. Instruktionerna i det här avsnittet gäller bara när du skapar en egen SWF-fil som läser in filen air.swf.*

Kontrollera om körningsversionen är installerad

En SWF-fil kan kontrollera om körningsversionen är installerad genom att anropa metoden `getStatus()` i filen air.swf som läses in från http://airdownload.adobe.com/air/browserapi/air.swf. Mer information finns i "Läsa in filen air.swf" på sidan 248.

När filen air.swf har lästs in kan SWF-filen anropa air.swf-filens `getStatus()`-metod så här:

```
var status:String = airSWF.getStatus();
```

Metoden `getStatus()` returnerar ett av följande strängvärden, baserat på statusen för körningsversionen på datorn:

Strängvärde	Beskrivning
`"available"`	Körningsversionen kan installeras på den här datorn men är för närvarande inte installerad.
`"unavailable"`	Körningsversionen kan inte installeras på den här datorn.
`"installed"`	Körningsversionen är installerad på den här datorn.

`getStatus()`-metoden returnerar ett fel om den obligatoriska versionen av Flash Player (version 9 uppdatering 3 eller senare i Windows och Mac OS, eller version 10 i Linux) inte är installerad i webbläsaren.

Kontrollera från en webbsida om ett AIR-program är installerat

En SWF-fil kan kontrollera om ett AIR-program (med matchande program-ID och utgivar-ID) är installerat genom att anropa metoden `getApplicationVersion()` i filen air.swf som läses in från http://airdownload.adobe.com/air/browserapi/air.swf. Mer information finns i "Läsa in filen air.swf" på sidan 248.

När filen air.swf har lästs in kan SWF-filen anropa air.swf-filens `getApplicationVersion()`-metod så här:

```
 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.
    }
}
```

Metoden `getApplicationVersion()` har följande parametrar:

Parametrar	Beskrivning
`appID`	Program-ID:t för programmet. Mer information finns i ”id” på sidan 221.
`pubID`	Utgivar-ID:t för programmet. Mer information finns i ”publisherID” på sidan 231. Om programmet i fråga inte har ett utgivar-ID anger du en tom sträng (“”) som parametern `pubID`.
`callback`	En callback-funktion som fungerar som hanterarfunktion. Metoden getApplicationVersion() arbetar asynkront, och när den installerade versionen identifieras (eller om det inte finns någon installerad version), anropas den här callback-metoden. Callback-metodens definition måste innehålla en parameter, en sträng, som ställs in på versionssträngen för det installerade programmet. Om programmet inte är installerat, skickas ett null-värde till funktionen, vilket visas i föregående kodexempel.

`getApplicationVersion()`-metoden returnerar ett fel om den obligatoriska versionen av Flash Player (version 9 uppdatering 3 eller senare i Windows och Mac OS, eller version 10 i Linux) inte är installerad i webbläsaren.

Obs! *Från och med AIR 1.5.3 används inte utgivar-ID längre. Utgivar-ID tilldelas inte längre automatiskt till ett program. För bakåtkompatibilitet kan program fortsätta att ange ett utgivar-ID.*

Installera ett AIR-program från webbläsaren

En SWF-fil kan installera ett AIR-program genom att anropa metoden `installApplication()` i filen air.swf som läses in från `http://airdownload.adobe.com/air/browserapi/air.swf`. Mer information finns i ”Läsa in filen air.swf” på sidan 248.

När filen air.swf har lästs in kan SWF-filen anropa air.swf-filens `installApplication()`-metod så här:

```
 var url:String = "http://www.example.com/myApplication.air";
var runtimeVersion:String = "1.0";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.installApplication(url, runtimeVersion, arguments);
```

Metoden `installApplication()` installerar angivet program på användarens dator. Den här metoden har följande parametrar:

Parameter	Beskrivning
`url`	En sträng som definierar URL:en för AIR-filen som ska installeras. Du måste använda en absolut URL-sökväg, inte en relativ.
`runtimeVersion`	En sträng som anger vilken körningsversion (till exempel ”1.0”) som krävs för programmet som installeras.
`argument`	En array med argument som ska skickas till programmet om det startar vid installationen. Endast alfanumeriska tecken kan identifieras i argumenten. Om du vill överföra andra värden bör du använda ett annat kodningsschema. Programmet startar vid installationen om elementet `allowBrowserInvocation` ställs in på `true` i programbeskrivningsfilen. (Mer information om programbeskrivningsfilen finns i ”AIR-programbeskrivningsfiler” på sidan 200.) Om programmet startar på grund av en sömlös installation från webbläsaren (där användaren väljer att starta vid installation), skickar programmets NativeApplication-objekt ett BrowserInvokeEvent-objekt bara om argument skickas. Mer information finns i ”Starta ett installerat AIR-program från webbläsaren” på sidan 251.

Metoden `installApplication()` fungerar bara när den anropas i händelsehanteraren för en användarhändelse, till exempel ett musklick.

`installApplication()`-metoden returnerar ett fel om den obligatoriska versionen av Flash Player (version 9 uppdatering 3 eller senare i Windows och Mac OS, eller version 10 i Linux) inte är installerad i webbläsaren.

För att installera en uppdaterad version av ett program i Mac OS måste användaren ha tillräcklig behörighet för att installera i programkatalogen (och administrativa behörigheter om programmet uppdaterar körningsversionen). I Windows måste en användare ha administrativ behörighet.

Du kan också anropa metoden `getApplicationVersion()` för filen air.swf om du vill kontrollera om ett program redan är installerat. Du kan anropa den här metoden innan programmet börjar installeras eller efter att installationen har startat. Mer information finns i ”Kontrollera från en webbsida om ett AIR-program är installerat” på sidan 249. När programmet körs kan det kommunicera med SWF-innehållet i webbläsaren genom att använda klassen LocalConnection. Mer information finns i Kommunicera med andra Flash Player- och AIR-instanser (för ActionScript-utvecklare) eller Communicating with other Flash Player and AIR instances (för HTML-utvecklare).

Starta ett installerat AIR-program från webbläsaren

Om du vill använda funktionen för webbläsaranrop (så att programmet kan startas från webbläsaren) måste programbeskrivningsfilen för målprogrammet innehålla följande sträng:

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Mer information om programbeskrivningsfilen finns i ”AIR-programbeskrivningsfiler” på sidan 200.

En SWF-fil i webbläsaren kan starta ett AIR-program genom att anropa metoden `launchApplication()` i filen air.swf som läses in från http://airdownload.adobe.com/air/browserapi/air.swf. Mer information finns i ”Läsa in filen air.swf” på sidan 248.

När filen air.swf har lästs in kan SWF-filen anropa air.swf-filens `launchApplication()`-metod så här:

```
var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.launchApplication(appID, pubID, arguments);
```

Metoden `launchApplication()` definieras på den översta nivån i filen air.swf (som läses in i programdomänen för användargränssnittets SWF-fil). När den här metoden anropas startar AIR angivet program (om det är installerat och webbläsaranrop tillåts, via inställningen `allowBrowserInvocation` i programbeskrivningsfilen). Metoden har följande parametrar:

Parameter	Beskrivning
`appID`	Program-ID:t för programmet som ska starta. Mer information finns i "id" på sidan 221.
`pubID`	Utgivar-ID:t för programmet som ska starta. Mer information finns i "publisherID" på sidan 231. Om programmet i fråga inte har ett utgivar-ID anger du en tom sträng ("") som parametern `pubID`.
`argument`	En array med argument som ska skickas till programmet. Objektet NativeApplication för programmet skickar en BrowserInvokeEvent-händelse som har egenskapen arguments inställd på den här arrayen. Endast alfanumeriska tecken kan identifieras i argumenten. Om du vill överföra andra värden bör du använda ett annat kodningsschema.

Metoden `launchApplication()` fungerar bara när den anropas i händelsehanteraren för en användarhändelse, till exempel ett musklick.

`launchApplication()`-metoden returnerar ett fel om den obligatoriska versionen av Flash Player (version 9 uppdatering 3 eller senare i Windows och Mac OS, eller version 10 i Linux) inte är installerad i webbläsaren.

Om elementet `allowBrowserInvocation` ställs in på `false` i programbeskrivningsfilen har ett anrop till metoden `launchApplication()` ingen effekt.

Innan användargränssnittet som startar programmet visas, kanske du vill anropa metoden `getApplicationVersion()` i filen air.swf. Mer information finns i "Kontrollera från en webbsida om ett AIR-program är installerat" på sidan 249.

När programmet anropas via funktionen för webbläsaranrop skickar programmets NativeApplication-objekt ett BrowserInvokeEvent-objekt. Mer information finns i Anropa ett AIR-program från webbläsaren (för ActionScript-utvecklare) eller Invoking an AIR application from the browser (för HTML-utvecklare).

Om du använder funktionen för webbläsaranrop måste du tänka på säkerheten. Dessa konsekvenser beskrivs i Anropa ett AIR-program från webbläsaren (for ActionScript developers) och Invoking an AIR application from the browser (för HTML-utvecklare).

När programmet körs kan det kommunicera med SWF-innehållet i webbläsaren genom att använda klassen LocalConnection. Mer information finns i Kommunicera med andra Flash Player- och AIR-instanser (för ActionScript-utvecklare) eller Communicating with other Flash Player and AIR instances (för HTML-utvecklare).

Obs! *Från och med AIR 1.5.3 används inte utgivar-ID längre. Utgivar-ID tilldelas inte längre automatiskt till ett program. För bakåtkompatibilitet kan program fortsätta att ange ett utgivar-ID.*

Kapitel 17: Uppdatera AIR-program

Användare kan installera eller uppdatera ett AIR-program genom att dubbelklicka på en AIR-fil på datorn eller i en webbläsare (med funktionen för sömlös installation). Installationsprogrammet för Adobe® AIR® hanterar installationen och varnar användaren om ett befintligt program uppdateras.

Du kan också låta ett installerat program uppdatera sig själv till en ny version med klassen Updater. (Ett installerat program kan detektera att det finns en ny version att hämta och installera.) Klassen Updater innehåller en `update()`-metod med vilken du kan peka på en AIR-fil på användarens dator och uppdatera till den versionen. Ditt program måste paketeras som en AIR-fil om klassen Updater ska kunna användas. Program som paketeras som systemspecifika körbara filer eller paket bör använda de uppdateringsfunktioner som den aktuella plattformen har.

Både program-id och utgivar-id för en uppdaterings-AIR-fil måste matcha programmet som ska uppdateras. Utgivar-ID:et kommer från signeringscertifikatet. Både uppdateringen och programmet som ska uppdateras måste vara signerade med samma certifikat.

I AIR 1.5.3 och senare inkluderas elementet `<publisherID>` i programbeskrivningsfilen. Du måste använda elementet om versioner av ditt program har utvecklats med AIR 1.5.2 eller tidigare. Du hittar mer information i avsnittet om ”publisherID” på sidan 231.

Från och med AIR 1.1 kan du flytta ett program så att det går att använda ett nytt kodsigneringscertifikat. Om ett program ska flyttas för att använda en ny signatur, innebär detta att uppdaterings-AIR-filen måste signeras med både det nya och det ursprungliga certifikatet. Certifikatflyttning är en envägsprocess. Efter flytten kommer programmet endast att känna igen AIR-filer som är signerade med det nya certifikatet (eller med båda certifikaten) som uppdateringar till en befintlig installation.

Det kan vara komplicerat att hantera uppdateringar av program. AIR 1.5 inkluderar det nya *uppdateringsramverket för AdobeAIR-program*. Det här ramverket innehåller API:er som utvecklarna kan använda för att tillhandahålla praktiska uppdateringsfunktioner i AIR-program.

Du kan använda certifikatflytt för att byta från ett självsignerat certifikat till ett kommersiellt kodsigneringscertifikat, eller för att byta från ett självsignerat eller kommersiellt certifikat till ett annat. Om du inte flyttar certifikatet, måste befintliga användare avinstallera den aktuella versionen av programmet innan de installerar den nya versionen. Mer information finns under ”Byta certifikat” på sidan 190.

Det är en god vana att inkludera en uppdateringsmekanism i programmet. Om du skapar en ny version av programmet kommer uppdateringsmekanismen att uppmana användaren att installera den nya versionen.

AIR-programmets installationsfil skapar loggfiler när ett AIR-program installeras, uppdateras eller tas bort. Du kan studera loggarna för att fastställa orsaken till eventuella installationsproblem. Läs mer i Installation logs.

Obs! *Nya versioner av Adobe AIR kan inkludera uppdaterade versioner av WebKit. En uppdaterad version av WebKit kan eventuellt leda till oväntade ändringar i HTML-innehållet i ett distribuerat AIR-program. Ändringarna kanske gör att du måste uppdatera programmet. En uppdateringsmekanism kan meddela användaren att det finns en ny version av programmet. Mer information finns i Om HTML-miljön (för ActionScript-utvecklare) eller i About the HTML environment (för HTML-utvecklare).*

Om att uppdatera program

Klassen Updater (i paketet flash.desktop) innehåller en metod, `update()`, som du kan använda för att uppdatera det program som körs för tillfället till en annan version. Om användaren till exempel har en version av AIR-filen ("Sample_App_v2.air") på skrivbordet, uppdaterar följande kod programmet.

ActionScript-exempel:

```
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-exempel:

```
var updater = new air.Updater();
var airFile = air.File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version = "2.01";
updater.update(airFile, version);
```

Innan ett program använder klassen Updater, måste användaren eller programmet hämta den uppdaterade versionen av AIR-filen till datorn. Mer information finns i avsnittet ”Hämta en AIR-fil till användarens dator” på sidan 256.

Resultat av ett anrop till metoden Updater.update()

När ett program i körningsmiljön anropar metoden `update()`, stänger körningsmiljön ned programmet och försöker sedan att installera den nya versionen från AIR-filen. Körningsmiljön kontrollerar att det program-id och det utgivar-id som anges i AIR-filen matchar program-id och utgivar-id för det program som anropar metoden `update()`. (Mer information om program-id och utgivar-id finns under ”AIR-programbeskrivningsfiler” på sidan 200.) Den kontrollerar också att versionssträngen matchar den `versions` sträng som skickas in i metoden `update()`. Om installationen genomförs, startar körningsmiljön den nya versionen av programmet. I annat fall (om det inte går att genomföra installationen) öppnar körningsmiljön den befintliga versionen (för-installationsversionen) av programmet.

För att installera en uppdaterad version av ett program i Mac OS måste användaren ha tillräcklig behörighet för att installera i programkatalogen. I Windows och Linux måste användaren ha administrativ behörighet.

Om den uppdaterade programversionen kräver en uppdaterad körningsmiljöversion installeras den nya körningsmiljöversionen. Användaren måste ha administrationsbehörighet för datorn för att kunna uppdatera körtiden.

När ett program testas med ADL, orsakar ett anrop av metoden `update()` ett körningsundantag.

Om versionssträngen

Den sträng som anges som `version`-parameter till metoden `update()` måste matcha strängen i elementet `version` eller `versionNumber` i programbeskrivningsfilen för den AIR-fil som ska installeras. Av säkerhetsskäl måste parametern `version` anges. Genom att begära att versionsnumret i AIR-filen kontrolleras i programmet, kommer inte en äldre version att installeras av misstag. (En äldre version av programmet kan innehålla en säkerhetsrisk som har åtgärdats i det program som är installerat för närvarande.) Programmet bör också jämföra versionssträngen i AIR-filen med versionssträngen i det installerade programmet för att förhindra nedgraderingsattacker.

I tidigare versioner än AIR 2.5 kan versionssträngen ha vilket format som helst. Strängen kan till exempel ha formatet "2.01" eller "version 2". I AIR 2.5 och senare versioner måste versionssträngen vara en sekvens med högst tre tresiffriga nummer, åtskilda med punkter. Till exempel är ”.0”, ”1.0” och ”67.89.999” alla giltiga versionsnummer. Du bör kontrollera uppdateringens versionssträng innan du uppdaterar programmet.

Om ett Adobe AIR-program hämtar en AIR-fil via webben, är det god praxis att ha en mekanism för att låta webbtjänsten signalera till Adobe AIR-programmet vilken version det är som hämtas. Programmet kan sedan använda denna sträng som `version`-parameter till metoden `update()`. Om AIR-filen hämtas på något annat sätt som gör att dess version är okänd, kan AIR-programmet undersöka AIR-filen för att avgöra dess versionsinformation. (En AIR-fil är ett ZIP-komprimerat arkiv, och programbeskrivningsfilen är den andra posten i arkivet.)

Mer information om programbeskrivningsfilen finns i ”AIR-programbeskrivningsfiler” på sidan 200.

Signeringsarbetsflöde för uppdateringar

Att publicera ad hoc-uppdateringar komplicerar arbetet med att hantera flera programversioner, och det blir också svårare att hålla reda på när alla olika certifikat upphör att gälla. Certifikat kan upphöra att gälla innan du kan publicera en uppdatering.

I Adobe AIR-miljön hanteras en programuppdatering som publicerats utan en flyttningssignatur som ett nytt program. Användarna måste avinstallera det aktuella AIR-programmet innan de kan installera programuppdateringen.

Du löser det här problemet genom att överföra varje uppdaterat program med det senaste certifikatet till en separat distributionsadress. Inkludera en funktion som påminner dig om att använda flyttningssignaturer när ditt certifikat är inom 180-dagarsfristen. Mer information finns i ”Signera en uppdaterad version av ett AIR-program” på sidan 195.

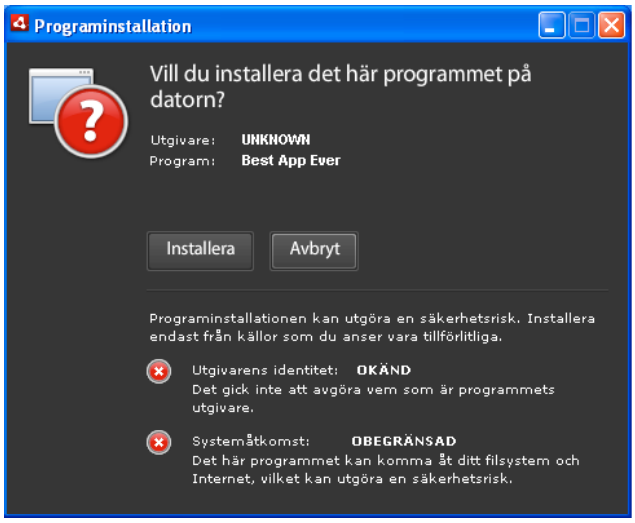
Avsnittet ”ADT-kommandon” på sidan 162 innehåller information om hur du använder signaturer.

Utför följande åtgärder för att effektivisera arbetet med att använda flyttningssignaturer:

- Överför varje uppdaterat program till en separat distributionsadress.
- Överför XML-beskrivningsfilen för uppdateringen och det senaste certifikatet för uppdateringen till samma adress.
- Signera det uppdaterade programmet med det senaste certifikatet.
- Använd en flyttningssignatur på det uppdaterade programmet med det certifikat som användes för att signera den tidigare versionen, som finns på en annan adress.

Skapa ett anpassat användargränssnitt för programuppdatering

AIR innehåller ett standardgränssnitt för uppdatering:



Detta gränssnitt används alltid den första gången en användare installerar en version av ett program på datorn. Du kan dock definiera ett eget gränssnitt som ska användas efterföljande gånger. Om ett anpassat uppdateringsgränssnitt definieras i programmet använder du ett `customUpdateUI`-element i programbeskrivningsfilen för det installerade programmet:

```
<customUpdateUI>true</customUpdateUI>
```

När programmet är installerat och användaren öppnar en AIR-fil med ett program-id och ett utgivar-id som matchar det installerade programmet, öppnar körningsmiljön programmet istället för standardinstallationsprogrammet för AIR. Läs mer i avsnittet om ”customUpdateUI” på sidan 211.

Programmet kan när det körs (när objektet `NativeApplication.nativeApplication` skickar en `load`-händelse) avgöra om det ska uppdateras eller inte (med klassen Updater). Om programmet ska uppdateras, visas dess egna installationsgränssnitt (som skiljer sig från dess standardgränssnitt vid körning) för användaren.

Hämta en AIR-fil till användarens dator

För att det ska gå att använda klassen Updater, måste användaren eller programmet först spara en AIR-fil lokalt på användarens dator.

Obs! *AIR 1.5 inkluderar ett uppdateringsramverk som hjälper utvecklarna att tillhandahålla uppdateringsfunktioner i AIR-program. Det kan vara betydligt enklare att använda det här ramverket än att använda Update-klassens* `update()`*-metod direkt. Mer information finns i avsnittet ”Använda uppdateringsramverket” på sidan 260.*

Följande kod läser en AIR-fil från en URL (http://example.com/air/updates/Sample_App_v2.air) och sparar AIR-filen i programlagringskatalogen.

ActionScript-exempel:

```
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-exempel:

```
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.");
}
```

Mer information finns i:

- Arbetsflöde för att läsa och skriva filer (för ActionScript-utvecklare)
- Workflow for reading and writing files (för HTML-utvecklare)

Kontrollera för att se om ett program körs för första gången

När du har uppdaterat ett program kanske du vill att ett Komma igång- eller Välkommen-meddelande ska visas för användaren. När programmet startas görs en kontroll av om det körs för första gången, för att se om meddelandet ska visas.

Obs! *AIR 1.5 inkluderar ett uppdateringsramverk som hjälper utvecklarna att tillhandahålla uppdateringsfunktioner i AIR-program. Det här ramverket innehåller enkla metoder som du kan använda för att kontrollera om en version av ett program körs för första gången. Mer information finns i avsnittet "Använda uppdateringsramverket" på sidan 260.*

Ett sätt att göra detta är att spara en fil i programlagringskatalogen när programmet initieras. Varje gång programmet startas bör det göras en kontroll för att se om den filen finns. Om filen inte finns, körs programmet för första gången för den aktuelle användaren. Om filen finns, har programmet redan körts minst en gång. Om filen finns och innehåller ett versionsnummer som är äldre än det aktuella versionsnumret, vet du att användaren kör den nya versionen för första gången.

I följande Flex-exempel demonstreras konceptet:

```
<?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>
```

I följande exempel demonstreras konceptet i JavaScript:

```
<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>
```

Om programmet sparar data lokalt (till exempel i programlagringskatalogen) kan du vilja kontrollera för att se om det finns tidigare sparad data (från tidigare versioner) vid den första körningen.

Använda uppdateringsramverket

Det kan vara jobbigt att hantera uppdateringar av program. *Uppdateringsramverket för AdobeAIR-program* innehåller API:er som utvecklare kan använda för att skapa stabila uppdateringsfunktioner i AIR-program. AIR-uppdateringsramverket kan utföra följande uppgifter för utvecklare:

- Regelbundet söka efter uppdateringar baserat på ett intervall eller på begäran av användaren
- Hämta AIR-filer (uppdateringar) från en webbkälla
- Meddela användaren första gången en ny version körs
- Bekräfta att användaren vill söka efter uppdateringar
- Visa information om den nya uppdateringsversionen för användaren
- Visa information om hämtningsförlopp och fel för användaren

AIR-uppdateringsramverket inkluderar ett exempelanvändargränssnitt för ditt program. Den förser användaren med grundläggande information och konfigurationsalternativ för programuppdateringar. Programmet kan också definiera ett anpassat användargränssnitt för användning med uppdateringsramverket.

Med AIR-uppdateringsramverket kan du lagra information om uppdateringsversionen för ett AIR-program i vanliga XML-konfigurationsfiler. I de flesta program kan du genom att skapa dessa konfigurationsfiler med grundläggande kod tillhandahålla bra uppdateringsfunktioner för slutanvändaren.

Även om du inte använder uppdateringsramverket innehåller Adobe AIR en Updater-klass, som kan användas med AIR-program för att uppgradera till nya versioner. Med hjälp av klassen Updater kan ett program uppgradera till en version som finns i en AIR-fil på användarens dator. Uppgraderingshanteringen kan emellertid omfatta mer än att bara uppdatera programmet baserat på en lokalt lagrad AIR-fil.

Ramverksfiler för AIR-uppdatering

Ramverksfiler för AIR-uppdatering finns i katalogen frameworks/libs/air i AIR 2 SDK. Där hittar du följande filer:

- applicationupdater.swc – definierar basfunktioner för uppdateringsbiblioteket som används i ActionScript. För denna version används inget användargränssnitt.
- applicationupdater.swf – definierar basfunktioner för uppdateringsbiblioteket som används i JavaScript. För denna version används inget användargränssnitt.
- applicationupdater_ui.swc – definierar en Flex 4-version för uppdateringsbibliotekets grundläggande funktioner, och inkluderar ett användargränssnitt som kan användas för att visa uppdateringsalternativ i programmet.
- applicationupdater_ui.swf – definierar en JavaScript-version för uppdateringsbibliotekets grundläggande funktioner, och inkluderar ett användargränssnitt som kan användas för att visa uppdateringsalternativ i programmet.

Mer information finns i dessa avsnitt:

- ”Konfigurera Flex-utvecklingsmiljön” på sidan 261
- ”Inkludera ramverksfiler i ett HTML-baserat AIR-program” på sidan 261
- ”Grundläggande exempel: Använda ApplicationUpdaterUI-versionen” på sidan 262

Konfigurera Flex-utvecklingsmiljön

SWC-filerna i katalogen frameworks/libs/air för AIR 2 SDK används för att definiera klasser som kan användas vid Flex- och Flash-utveckling.

Om du vill använda uppdateringsramverket vid kompilering med Flex SDK inkluderar du filen ApplicationUpdater.swc eller filen ApplicationUpdater_UI.swc i anropet till amxmlc-kompilatorn. I följande exempel läser kompilatorn in filen ApplicationUpdater.swc i underkatalogen lib i Flex SDK-katalogen:

```
amxmlc -library-path+=lib/ApplicationUpdater.swc  -- myApp.mxml
```

I följande exempel läser kompilatorn in filen ApplicationUpdater_UI.swc i underkatalogen lib i katalogen Flex SDK:

```
amxmlc -library-path+=lib/ApplicationUpdater_UI.swc  -- myApp.mxml
```

Om du använder Flash Builder lägger du till SWC-filen på fliken Library Path under inställningarna för Flex Build Path i dialogrutan Properties.

Kopiera SWC-filerna till den katalog som du ska referera till i amxmlc-kompilatorn (med hjälp av Flex SDK) eller Flash Builder.

Inkludera ramverksfiler i ett HTML-baserat AIR-program

Katalogen frameworks/html i uppdateringsramverket innehåller följande SWF-filer:

- applicationupdater.swf – Definierar uppdateringsbibliotekets grundläggande funktioner, utan något användargränssnitt
- applicationupdater_ui.swf – Definierar uppdateringsbibliotekets grundläggande funktioner och inkluderar ett användargränssnitt, som kan användas för att visa uppdateringsalternativ i programmet

JavaScript-kod i AIR-program kan använda klasser som definieras i SWF-filer.

Om du vill använda uppdateringsramverket lägger du till filen applicationupdater.swf eller filen applicationupdater_ui.swf i programkatalogen (eller i en underkatalog). I HTML-filen som ska använda ramverket (i JavaScript-kod) lägger du sedan till en `script`-tagg som läser in filen:

```
<script src="applicationUpdater.swf" type="application/x-shockwave-flash"/>
```

Du kan också använda den här `script`-taggen för att läsa in filen applicationupdater_ui.swf:

```
<script src="applicationupdater_ui.swf" type="application/x-shockwave-flash"/>
```

API:et som definieras i dessa två filer beskrivs närmare i resten av det här dokumentet.

Grundläggande exempel: Använda ApplicationUpdaterUI-versionen

ApplicationUpdaterUI-versionen av uppdateringsramverket tillhandahåller ett enkelt gränssnitt som du lätt kan använda i ditt program. Här följer ett grundläggande exempel.

Börja med att skapa ett AIR-program som anropar uppdateringsramverket:

1 Om programmet är ett HTML-baserat AIR-program läser du in filen applicationupdaterui.swf:

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

2 Skapa ett ApplicationUpdaterUI-objekt i programlogiken för AIR-programmet.

Använd följande kod i ActionScript:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

Använd följande kod i JavaScript:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

Du kan lägga till den här koden i en initieringsfunktion som körs när programmet har lästs in.

3 Skapa en textfil vid namn updateConfig.xml och lägg till följande i filen:

```
<?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>
```

Ändra `URL`-elementet för updateConfig.xml-filen så att det matchar platsen för uppdateringsbeskrivningsfilen på webbservern (se nästa procedur).

`delay` motsvarar antalet dagar mellan uppdateringskontrollerna.

4 Lägg till updateConfig.xml-filen i AIR-programmets projektkatalog.

5 Låt updater-objektet referera till updateConfig.xml-filen och anropa objektets `initialize()`-metod.

Använd följande kod i ActionScript:

```
appUpdater.configurationFile = new File("app:/updateConfig.xml");
appUpdater.initialize();
```

Använd följande kod i JavaScript:

```
appUpdater.configurationFile = new air.File("app:/updateConfig.xml");
appUpdater.initialize();
```

6 Skapa en andra version av AIR-programmet som har en annan version än det första programmet. (Versionen anges i programbeskrivningsfilen, i `version`-elementet.)

Placera sedan uppdateringsversionen av AIR-programmet på webbservern:

1 Placera uppdateringsversionen av AIR-filen på webbservern.

2 Skapa en textfil med namnet updateDescriptor.2.5.xml och lägg till följande innehåll i filen:

```
<?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>
```

Redigera `versionNumber`, `URL` och `description` i filen updateDescriptor.xml så att de matchar AIR-uppdateringsfilen. Det här uppdateringsbeskrivningsformatet används av program som använder det uppdateringsramverk som ingår i AIR 2.5 SDK (och senare).

3 Skapa en textfil med namnet updateDescriptor.1.0.xml och lägg till följande innehåll i filen:

```
<?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>
```

Redigera `version`, `URL` och `description` i filen updateDescriptor.xml så att de matchar AIR-uppdateringsfilen. Det här uppdateringsbeskrivningsformatet används av program som använder det uppdateringsramverk som ingår i AIR 2 SDK (och tidigare).

Obs! *Du behöver bara skapa den här andra uppdateringsbeskrivningsfilen om du vill ha stöd för uppdatering av program som skapats i versioner äldre än AIR 2.5.*

4 Lägg till filen updateDescriptor.2.5.xml och updateDescriptor.1.0.xml i samma webbserverkatalog som innehåller AIR-uppdateringsfilen.

Det här är ett enkelt exempel, men tillhandahåller tillräckliga uppdateringsfunktioner för många program. I resten av det här dokumentet lär du dig hur du använder uppdateringsramverket som det passar dig bäst.

Om du vill har fler exempel på hur du använder uppdateringsramverket ska du titta på följande exempelprogram i Adobe AIR Developer Center:

- Uppdatera Framework i ett Flash-baserat program
  (http://www.adobe.com/go/learn_air_qs_update_framework_flash_se)

Uppdatera till AIR 2.5

Eftersom reglerna för tilldelning av versionsnummer i program har ändrats i AIR 2.5 kan uppdateringsramverket i AIR 2 inte tolka versionsinformationen i en AIR 2.5-programbeskrivning. Den här inkompatibiliteten innebär att du måste uppdatera ditt program så att det använder det nya uppdateringsramverket INNAN du uppdaterar ditt program till AIR 2.5 SDK. Det krävs alltså TVÅ uppdateringar för att uppdatera ditt program till AIR 2.5 eller senare från äldre versioner än AIR 2.5. Den första uppdateringen måste använda AIR 2-namnutrymmet och innehålla biblioteket för AIR 2.5-uppdateringsramverket (du kan fortfarande skapa programpaketet med AIR 2.5 SDK). Den andra uppdateringen kan använda AIR 2.5-namnutrymmet och innehålla programmets nya funktioner.

Du kan också använda den mellanliggande uppdateringen enbart för att uppdatera ditt AIR 2.5-program direkt med AIR-klassen Updater.

Följande exempel visar hur du uppdaterar ett program från version 1.0 till 2.0. Version 1.0 använder det gamla 2.0-namnutrymmet. Version 2.0 använder 2.5-namnutrymmet och de nya funktionerna implementeras med API:erna i AIR 2.5.

1 Skapa en mellanliggande version av programmet, version 1.0.1, baserat på version 1.0 av programmet.

   a Använd programuppdateringsramverket i AIR 2.5 när du skapar programmet.

   Obs! *Använd* `applicationupdater.swc` *eller* `applicationupdater_ui.swc` *för AIR-program som bygger på Flash-teknik och* `applicationupdater.swf` *eller* `applicationupdater_ui.swf` *för HTML-baserade AIR-program.*

   b Skapa en uppdateringsbeskrivningsfil för version 1.0.1 genom att använda det gamla namnutrymmet och den version som visas nedan:

```
<?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 Skapa version 2.0 av programmet med API:erna i AIR 2.5 och 2.5-namnutrymmet.

3 Skapa en uppdateringsbeskrivning för att uppdatera programmet från version 1.0.1 till version 2.0.

```
<?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>
```

Definiera uppdateringsbeskrivningsfiler och placera AIR-filen på webbservern

När du använder AIR-uppdateringsramverket anger du grundläggande information om den tillgängliga uppdateringen i uppdateringsbeskrivningsfiler, som lagras på webbservern. En uppdateringsbeskrivningsfil är en vanlig XML-fil. Uppdateringsramverket i programmet använder den här filen för att kontrollera om en ny version har överförts.

Formatet för uppdateringsbeskrivningsfilen har ändrats i AIR 2.5. I det nya formatet används ett annat namnutrymme. Det ursprungliga namnutrymmet är ”http://ns.adobe.com/air/framework/update/description/1.0”. AIR 2.5-namnutrymmet är ”http://ns.adobe.com/air/framework/update/description/2.5”.

AIR-program som skapats i tidigare version än AIR 2.5 kan bara läsa uppdateringsbeskrivningen för version 1.0. AIR-program som skapats med uppdateringsramverket i AIR 2.5 eller senare kan bara läsa uppdateringsbeskrivningen för version 2.5. På grund av den här versionsinkompatibiliteten måste du ofta skapa två uppdateringsbeskrivningsfiler. Uppdateringslogiken i AIR 2.5-versioner av ditt program måste hämta en uppdateringsbeskrivning som använder det nya formatet. Äldre versioner av ditt AIR-program måste fortsätta att använda det ursprungliga formatet. Båda filerna måste ändras för varje uppdatering som du släpper (tills du slutar att ha stöd för äldre versioner än AIR 2.5).

Uppdateringsbeskrivningsfilen innehåller följande information:

- `versionNumber` – Den nya versionen av AIR-programmet. Använd elementet `versionNumber` i uppdateringsbeskrivningar som används för att uppdatera AIR 2.5-program. Värdet måste vara samma sträng som används i elementet `versionNumber` i den nya AIR-programbeskrivningsfilen. Om versionsnumret i uppdateringsbeskrivningsfilen inte matchar AIR-uppdateringsfilens versionsnummer genererar uppdateringsramverket ett undantagsfel.
- `version` – Den nya versionen av AIR-programmet. Använd elementet `version` i uppdateringsbeskrivningar som används för att uppdatera program som skapats före AIR 2.5. Värdet måste vara samma sträng som används i elementet `version` i den nya AIR-programbeskrivningsfilen. Om versionen i uppdateringsbeskrivningsfilen inte matchar AIR-uppdateringsfilens version genererar uppdateringsramverket ett undantag.
- `versionLabel` – Den läsbara versionssträng som ska visas för användarna. `versionLabel` är valfritt, men kan bara anges i version 2.5-uppdateringsbeskrivningsfiler. Använd det om du använder en `versionLabel` i programbeskrivningen och ge dem då samma värde.
- `url` – Platsen för AIR-uppdateringsfilen. Det här är filen som innehåller uppdateringsversionen av AIR-programmet.
- `description` – Information om den nya versionen. Den här informationen kan visas för användaren under uppdateringsprocessen.

Elementen `version` och `url` är obligatoriska. Elementet `description` är valfritt.

Här följer ett exempel på en 2.5-uppdateringsbeskrivningsfil:

```
<?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>
```

Och här är ett exempel på en 1.0-uppdateringsbeskrivningsfil:

```
<?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>
```

Om du vill definiera `description`-taggen på flera språk använder du flera `text`-element som definierar ett `lang`-attribut:

```
<?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>
```

Placera uppdateringsbeskrivningsfilen och AIR-uppdateringsfilen på webbservern.

Katalogen templates som inkluderas med uppdateringsbeskrivningen innehåller exempel på uppdateringsbeskrivningsfiler. Dessa omfattar versioner för ett och flera språk.

Skapa ett updater-objekt

När du har läst in AIR-uppdateringsramverket i koden (mer information finns i avsnittet ”Konfigurera Flex-utvecklingsmiljön” på sidan 261 och ”Inkludera ramverksfiler i ett HTML-baserat AIR-program” på sidan 261) måste du skapa ett updater-objekt som i nedanstående exempel.

ActionScript-exempel:

```
var appUpdater:ApplicationUpdater = new ApplicationUpdater();
```

JavaScript-exempel:

```
var appUpdater = new runtime.air.update.ApplicationUpdater();
```

I den här koden används ApplicationUpdater-klassen (som inte tillhandahåller något användargränssnitt). Om du vill använda klassen ApplicationUpdaterUI (som tillhandahåller ett användargränssnitt) använder du följande.

ActionScript-exempel:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

JavaScript-exempel:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

I de återstående kodexemplen i det här dokumentet förutsätts det att du har skapat ett updater-objekt vid namn `appUpdater`.

Konfigurera uppdateringsinställningarna

Både ApplicationUpdater och ApplicationUpdaterUI kan konfigureras via en konfigurationsfil som levereras med programmet eller via ActionScript eller JavaScript i programmet.

Definiera uppdateringsinställningar i en XML-konfigurationsfil

Konfigurationsfilen för uppdateringen är en XML-fil. Filen kan innehålla följande element:

- `updateURL` – En sträng. Representerar platsen för uppdateringsbeskrivningen på fjärrservern. Alla giltiga URLRequest-platser tillåts. Du måste definiera egenskapen `updateURL`, antingen via konfigurationsfilen eller via ett skript (mer information finns i avsnittet ”Definiera uppdateringsbeskrivningsfiler och placera AIR-filen på webbservern” på sidan 264). Du måste definiera den här egenskapen innan du använder updater-objektet (innan `initialize()`-metoden för updater-objektet anropas, vilket beskrivs i avsnittet ”Initiera uppdateringsramverket” på sidan 269).
- `delay` – Ett tal. Representerar ett tidsintervall som anger antalet dagar mellan uppdateringskontrollerna (värden som `0,25` tillåts). Värdet 0 (som är standardvärdet) anger att updater-objektet inte utför någon automatisk regelbunden kontroll.

Konfigurationsfilen för ApplicationUpdaterUI kan innehålla följande element förutom `updateURL`- och `delay`-elementen:

- `defaultUI`: En lista med `dialog`-element. Varje `dialog`-element har ett `name`-attribut som motsvarar en dialogruta i användargränssnittet. Varje `dialog`-element har ett `visible`-attribut som definierar om dialogrutan är synlig. Standardvärdet är `true`. `name`-attributet kan ha följande värden:
  - `"checkForUpdate"` – Motsvarar dialogrutorna Check for Update, No Update och Update Error

- `"downloadUpdate"` – Motsvarar dialogrutan Download Update
- `"downloadProgress"` – Motsvarar dialogrutorna Download Progress och Download Error
- `"installUpdate"` – Motsvarar dialogrutan Install Update
- `"fileUpdate"` – Motsvarar dialogrutorna File Update, File No Update och File Error

- `"unexpectedError"` – Motsvarar dialogrutan Unexpected Error

  Om värdet är `false` visas inte motsvarande dialogruta som en del av uppdateringsproceduren.

Här följer ett exempel på konfigurationsfilen för ApplicationUpdater-ramverket:

```
<?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>
```

Här är ett exempel på konfigurationsfilen för ApplicationUpdaterUI-ramverket, som inkluderar en definition för `defaultUI`-elementet:

```
<?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>
```

Peka på egenskapen `configurationFile` för att ange platsen för filen:

ActionScript-exempel:

```
appUpdater.configurationFile = new File("app:/cfg/updateConfig.xml");
```

JavaScript-exempel:

```
appUpdater.configurationFile = new air.File("app:/cfg/updateConfig.xml");
```

Katalogen templates i uppdateringsramverket innehåller exempelkonfigurationsfilen config-template.xml.

Definiera uppdateringsinställningar (ActionScript- eller JavaScript-kod)

Dessa konfigurationsparametrar kan också anges med hjälp av kod i programmet på följande sätt:

```
appUpdater.updateURL = " http://example.com/updates/update.xml";
appUpdater.delay = 1;
```

Egenskaperna för updater-objektet är `updateURL` och `delay`. Dessa egenskaper definierar samma inställningar som elementen `updateURL` och `delay` i konfigurationsfilen: URL:en för uppdateringsbeskrivningsfilen och intervallet för uppdateringskontroller. Om du anger en konfigurationsfil *och* inställningar i koden, prioriteras egenskaper som angetts i koden framför motsvarande inställningar i konfigurationsfilen.

Du måste definiera egenskapen `updateURL`, antingen via konfigurationsfilen eller via skript (mer information finns i avsnittet ”Definiera uppdateringsbeskrivningsfiler och placera AIR-filen på webbservern” på sidan 264) innan du använder updater-objektet (innan updater-objektets `initialize()`-metod anropas, vilket beskrivs i avsnittet ”Initiera uppdateringsramverket” på sidan 269).

ApplicationUpdaterUI-ramverket definierar också följande egenskaper för updater-objektet:

- `isCheckForUpdateVisible` – Motsvarar dialogrutorna Check for Update, No Update och Update Error
- `isDownloadUpdateVisible` – Motsvarar dialogrutan Download Update
- `isDownloadProgressVisible` – Motsvarar dialogrutorna Download Progress och Download Error
- `isInstallUpdateVisible` – Motsvarar dialogrutan Install Update
- `isFileUpdateVisible` – Motsvarar dialogrutorna File Update, File No Update och File Error
- `isUnexpectedErrorVisible` – Motsvarar dialogrutan Unexpected Error

Varje egenskap motsvarar en eller flera dialogrutor i ApplicationUpdaterUI-användargränssnittet. Varje egenskap är ett booleskt värde, med standardvärdet `true`. Om värdet är `false` visas inte motsvarande dialogrutor som en del av uppdateringsproceduren.

Dessa dialogruteegenskaper åsidosätter inställningarna i konfigurationsfilen för uppdateringen.

Uppdateringsprocessen

AIR-uppdateringsramverket slutför uppdateringsprocessen i följande steg:

1. Updater-objektets initieringsmetod kontrollerar om en uppdateringskontroll har utförts inom angivet tidsintervall (mer information finns i avsnittet ”Konfigurera uppdateringsinställningarna” på sidan 266). Om en uppdateringskontroll behöver köras fortsätter uppdateringsprocessen.
2. Updater-objektet hämtar och tolkar uppdateringsbeskrivningsfilen.
3. Updater-objektet hämtar AIR-uppdateringsfilen.
4. Updater-objektet installerar den uppdaterade versionen av programmet.

Updater-objektet skickar händelser när de olika stegen har slutförts. I ApplicationUpdater-versionen kan du avbryta händelserna som anger att ett steg i processen har slutförts. Om du avbryter någon av dessa händelser, avbryts nästa steg i processen. I ApplicationUpdaterUI-versionen visas en dialogruta där användaren kan välja att avbryta eller fortsätta varje steg i processen.

Om du avbryter händelsen kan du återuppta processen genom att anropa metoder för updater-objektet.

När ApplicationUpdater-versionen för updater-objektet avancerar genom uppdateringsprocessen registreras den aktuella statusen i en `currentState`-egenskap. Den här egenskapen är en sträng som kan ha något av följande värden:

- `"UNINITIALIZED"` – Updater-objektet har inte initierats.
- `"INITIALIZING"` – Updater-objektet initieras.
- `"READY"` – Updater-objektet har initierats.
- `"BEFORE_CHECKING"` – Updater-objektet har inte sökt efter uppdateringsbeskrivningsfilen än.
- `"CHECKING"` – Updater-objektet söker efter en uppdateringsbeskrivningsfil.
- `"AVAILABLE"` – Uppdateringsbeskrivningsfilen är tillgänglig.
- `"DOWNLOADING"` – Updater-objektet hämtar AIR-filen.
- `"DOWNLOADED"` – Updater-objektet har hämtat AIR-filen.
- `"INSTALLING"` – Updater-objektet installerar AIR-filen.
- `"PENDING_INSTALLING"` – Updater-objektet har initierats och det finns nya uppdateringar att installera.

En del metoder i updater-objektet körs bara om updater-objektet har en särskild status.

Initiera uppdateringsramverket

När du har angett konfigurationsegenskaperna (mer information finns i avsnittet ”Grundläggande exempel: Använda ApplicationUpdaterUI-versionen” på sidan 262) startar du uppdateringen genom att anropa `initialize()`-metoden:

```
appUpdater.initialize();
```

Den här metoden gör följande:

- Den initierar uppdateringsramverket och installerar synkront eventuella nya uppdateringar i bakgrunden. Metoden måste anropas under programstarten eftersom programmet kan startas om när den anropas.
- Den kontrollerar om det finns en uppdatering som har skjutits upp och installerar den.
- Om det uppstår ett fel under uppdateringen tar den bort uppdateringsfilen och versionsinformationen från programlagringsområdet.
- Om tidsintervallet har gått ut startar den uppdateringsprocessen. Annars startar den om timern.

När den här metoden anropas kan updater-objektet skicka följande händelser:

- `UpdateEvent.INITIALIZED` – Skickas när initieringen har slutförts.
- `ErrorEvent.ERROR` – Skickas om ett fel uppstår under initieringen.

Uppdateringsprocessen slutförs när `UpdateEvent.INITIALIZED`-händelsen har skickats.

När du anropar `initialize()`-metoden startar updater-objektet uppdateringsprocessen och slutför alla steg baserat på tidsfördröjningsinställningen (tidsintervallet). Du kan emellertid starta uppdateringsprocessen när du vill genom att anropa updater-objektets `checkNow()`-metod:

```
appUpdater.checkNow();
```

Den här metoden gör ingenting om uppdateringsprocessen redan körs. Annars startar den uppdateringsprocessen.

Updater-objektet kan skicka följande händelse som ett resultat av anropet till `checkNow()`-metoden:

- `UpdateEvent.CHECK_FOR_UPDATE`-händelsen strax innan det försöker hämta uppdateringsbeskrivningsfilen.

Om du avbryter `checkForUpdate`-händelsen kan du anropa updater-objektets `checkForUpdate()`-metod. (Mer information finns i nästa avsnitt.) Om du inte avbryter händelsen fortsätter uppdateringsprocessen att söka efter uppdateringsbeskrivningsfilen.

Hantera uppdateringsprocessen i ApplicationUpdaterUI-versionen

I ApplicationUpdaterUI-versionen kan användaren avbryta processen genom att klicka på Cancel i dialogrutorna i användargränssnittet. Du kan också avbryta uppdateringsprocessen genom att anropa ApplicationUpdaterUI-objektets `cancelUpdate()`-metod.

Du kan ange egenskaper för ApplicationUpdaterUI-objektet eller definiera element i konfigurationsfilen för uppdateringen om du vill ange vilka dialogrutealternativ som updater-objektet ska visa. Mer information finns i avsnittet ”Konfigurera uppdateringsinställningarna” på sidan 266.

Hantera uppdateringsprocessen i ApplicationUpdater-versionen

Du kan anropa `preventDefault()`-metoden för händelseobjekt som skickas av ApplicationUpdater-objektet om du vill avbryta steg i uppdateringsprocessen (mer information finns i avsnittet ”Uppdateringsprocessen” på sidan 268). Om du avbryter standardbeteendet kan du ange att ett meddelande ska visas där användaren tillfrågas om han eller hon vill fortsätta.

I följande avsnitt lär du dig hur du fortsätter uppdateringsprocessen när ett steg i processen har avbrutits.

Hämta och tolka uppdateringsbeskrivningsfilen

ApplicationUpdater-objektet skickar `checkForUpdate`-händelsen innan uppdateringsprocessen börjar, strax innan updater-objektet försöker hämta uppdateringsbeskrivningsfilen. Om du avbryter `checkForUpdate`-händelsens standardbeteende hämtas inte uppdateringsbeskrivningsfilen. Du kan anropa `checkForUpdate()`-metoden om du vill fortsätta uppdateringsprocessen:

```
appUpdater.checkForUpdate();
```

Om du anropar `checkForUpdate()`-metoden hämtar och tolkar updater-objektet uppdateringsbeskrivningsfilen asynkront. Som ett resultat av anropet till `checkForUpdate()`-metoden kan updater-objektet skicka följande händelser:

- `StatusUpdateEvent.UPDATE_STATUS` – Updater-objektet har hämtat och tolkat uppdateringsbeskrivningsfilen. Den här händelsen har följande egenskaper:
  - `available` – Ett booleskt värde. Använd värdet `true` om det finns en annan version än det aktuella programmets version. I annat fall använder du `false` (om versionerna är desamma).
  - `version` – En sträng. Versionen från uppdateringsfilens programbeskrivningsfil
  - `details` – En array. Om det inte finns några lokaliserade versioner av beskrivningen returnerar arrayen en tom sträng (`""`) som det första elementet och beskrivningen som det andra elementet.

    Om det finns flera versioner av beskrivningen (i uppdateringsbeskrivningsfilen) innehåller arrayen flera underarrayer. Varje array har två element: det första är en språkkod (t.ex. `"en"`) och det andra är motsvarande beskrivning (en sträng) för språket. Se ”Definiera uppdateringsbeskrivningsfiler och placera AIR-filen på webbservern” på sidan 264.
- `StatusUpdateErrorEvent.UPDATE_ERROR` – Det uppstod ett fel och updater-objektet kunde inte hämta eller tolka uppdateringsbeskrivningsfilen.

Hämta AIR-uppdateringsfilen

ApplicationUpdater-objektet skickar `updateStatus`-händelsen när updater-objektet har hämtat och tolkat uppdateringsbeskrivningsfilen. Som standard initieras hämtningen av uppdateringsfilen om den är tillgänglig. Om du avbryter standardbeteendet kan du fortsätta uppdateringsprocessen genom att anropa `downloadUpdate()`-metoden:

```
appUpdater.downloadUpdate();
```

Om du anropar den här metoden hämtar updater-objektet uppdateringsversionen av AIR-filen asynkront.

`downloadUpdate()`-metoden kan skicka följande händelser:

- `UpdateEvent.DOWNLOAD_START` – Anslutningen till servern har upprättats. Om ApplicationUpdaterUI-biblioteket används visar den här händelsen en dialogruta med en förloppsindikator över hämtningsförloppet.
- `ProgressEvent.PROGRESS` – Skickas regelbundet i takt med att filhämtningen avancerar.
- `DownloadErrorEvent.DOWNLOAD_ERROR` – Skickas om det inträffar ett fel vid anslutningen eller hämtningen av uppdateringsfilen. Skickas också för en ogiltig HTTP-status (t.ex. "404 – Det gick inte att hitta filen"). Den här händelsen har en `errorID`-egenskap, som är ett heltal som anger ytterligare felinformation. Ytterligare en `subErrorID`-egenskap kan innehålla mer felinformation.
- `UpdateEvent.DOWNLOAD_COMPLETE` – Updater-objektet har hämtat och tolkat uppdateringsbeskrivningsfilen. Om du inte avbryter den här händelsen fortsätter ApplicationUpdater-versionen att installera uppdateringsversionen. I ApplicationUpdaterUI-versionen visas en dialogruta där användaren kan välja att fortsätta.

Uppdatera programmet

ApplicationUpdater-objektet skickar `downloadComplete`-händelsen när uppdateringsfilen har hämtats. Om du avbryter standardbeteendet kan du fortsätta uppdateringsprocessen genom att anropa `installUpdate()`-metoden:

```
appUpdater.installUpdate(file);
```

Om du anropar den här metoden installerar updater-objektet en uppdateringsversion av AIR-filen. Metoden inkluderar en parameter, `file`, som är ett File-objekt som refererar till AIR-filen som ska användas som uppdatering.

ApplicationUpdater-objektet kan skicka `beforeInstall`-händelsen som ett resultat av ett anrop till `installUpdate()`-metoden:

- `UpdateEvent.BEFORE_INSTALL` – Skickas strax innan uppdateringen installeras. Ibland är det praktiskt att förhindra installationen av uppdateringen i det här läget, så att användaren kan slutföra pågående arbete innan uppdateringen fortsätter. Om Event-objektets `preventDefault()`-metod anropas skjuts installationen upp till nästa omstart, och ingen ny uppdateringsprocess kan startas. (Detta omfattar uppdateringar som resulterar från anrop till `checkNow()`-metoden eller de regelbundna kontrollerna.)

Installera från en godtycklig AIR-fil

Du kan anropa `installFromAIRFile()`-metoden om du vill att uppdateringsversionen ska installeras från en AIR-fil på användarens dator:

```
appUpdater.installFromAIRFile();
```

Med den här metoden installerar updater-objektet en uppdateringsversion av programmet från AIR-filen.

`installFromAIRFile()`-metoden kan skicka följande händelser:

- `StatusFileUpdateEvent.FILE_UPDATE_STATUS` – Skickas när ApplicationUpdater har validerat filen som skickats med `installFromAIRFile()`-metoden. Den här händelsen har följande egenskaper:
  - `available` – Använd värdet `true` om det finns en annan version än det aktuella programmets version. Annars använder du `false` (versionerna är samma).
  - `version` – Strängen som representerar den nya tillgängliga versionen.
  - `path` – Representerar uppdateringsfilens ursprungliga sökväg.

  Du kan avbryta den här händelsen om egenskapen för StatusFileUpdateEvent-objektet har värdet `true`. Om du avbryter händelsen avbryts uppdateringen. Anropa `installUpdate()`-metoden om du vill återuppta den avbrutna uppdateringen.
- `StatusFileUpdateErrorEvent.FILE_UPDATE_ERROR` – Det uppstod ett fel och updater-objektet kunde inte installera AIR-programmet.

Avbryta uppdateringsprocessen

Du kan anropa `cancelUpdate()`-metoden om du vill avbryta uppdateringsprocessen:

```
appUpdater.cancelUpdate();
```

Med den här metoden avbryts alla pågående hämtningar, eventuella ofullständiga hämtningar tas bort och timern för regelbundna kontroller startas om.

Metoden gör ingenting om updater-objektet initieras.

Lokalisera ApplicationUpdaterUI-gränssnittet

ApplicationUpdaterUI-klassen tillhandahåller ett standardanvändargränssnitt för uppdateringsprocessen. Användargränssnittet innehåller dialogrutor med alternativ för att starta processen, avbryta processen, samt för att utföra andra relaterade åtgärder.

Med uppdateringsbeskrivningsfilens `description`-element kan du definiera beskrivningen av programmet på flera språk. Använd flera `text`-element som definierar relevanta `lang`-attribut, som i följande exempel:

```
<?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>
```

Uppdateringsramverket använder den beskrivning som bäst överensstämmer med slutanvändarens lokaliseringskedja. Mer information finns i avsnittet Definiera uppdateringsbeskrivningsfilen och placera AIR-filen på webbservern.

Flex-utvecklare kan direkt lägga till ett nytt språk i `"ApplicationUpdaterDialogs"`-paketet.

JavaScript-utvecklare kan anropa updater-objektets `addResources()`-metod. Med den här metoden läggs ett nytt resurspaket för ett språk till dynamiskt. Resurspaketet innehåller lokaliserade strängar för ett språk. Dessa strängar används i olika textfält i dialogrutor.

JavaScript-utvecklare kan använda ApplicationUpdaterUI-klassens `localeChain`-egenskap för att definiera språkkedjan som används i användargränssnittet. Normalt sett använder bara JavaScript-utvecklare (HTML) den här egenskapen. Flex-utvecklare kan hantera språkkedjan med hjälp av ResourceManager.

Följande JavaScript-kod definierar exempelvis resurspaket för rumänska och ungerska:

```
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;
```

Mer information finns i beskrivningen av ApplicationUpdaterUI-klassens `addResources()`-metod i språkreferensen.

Kapitel 18: Visa källkod

Precis som en användare kan visa källkod för en HTLM-sida i en webbläsare kan han/hon visa källkoden för ett HTML-baserat AIR-program. I Adobe® AIR® SDK finns JavaScript-filen AIRSourceViewer.js som du kan använda i programmet för att visa källkoden för slutanvändarna.

Läsa in, konfigurera och öppna Source Viewer

Koden för Source Viewer finns i JavaScript-filen, AIRSourceViewer.js, som ingår i ramverkskatalogen för AIR SDK. Om du vill använda Source Viewer i ditt program ska du kopiera AIRSourceViewer.js till programmets projektkatalog och läsa in filen via en script-tagg i HTML-huvudfilen i programmet:

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

I AIRSourceViewer.js-filen definieras en klass, SourceViewer, som du kommer åt från JavaScript-koden genom att anropa `air.SourceViewer`.

SourceViewer-klassen definierar tre metoder: `getDefault()`, `setup()` och `viewSource()`.

Metod	Beskrivning
`getDefault()`	En statisk metod. Den returnerar en SourceViewer-instans som du kan använda för att anrop de andra metoderna.
`setup()`	Tillämpar konfigurationsinställningar för Source Viewer. Mer information finns i "Konfigurera Source Viewer" på sidan 273
`viewSource()`	Öppnar ett nytt fönster där användaren kan bläddra och öppna källfiler för värdprogrammet.

Obs! *Kod där Source Viewer används måste finnas i programmets säkerhetssandlåda (i en fil i programkatalogen).*

I följande JavaScript-kod instansieras ett Source Viewer-objeket och Source Viewer-fönstret öppnas med alla källfiler synliga:

```
var viewer = air.SourceViewer.getDefault();
viewer.viewSource();
```

Konfigurera Source Viewer

Metoden `config()` tilldelar givna inställningar till Source Viewer. Den här metoden har en parameter: `configObject`. Objekten `configObject` har egenskaper som definierar konfigurationsinställningar för Source Viewer. Egenskaperna är `default`, `exclude`, `initialPosition`, `modal`, `typesToRemove` och `typesToAdd`.

default

En sträng som definierar den relativa sökvägen till den första filen som ska visas i Source Viewer.

Med exempelvis följande JavaScript-kod öppnas Source Viewer-fönstret med filen index.html som den första filen:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.default = "index.html";
viewer.viewSource(configObj);
```

exclude
En strängmatris som definierar filer eller kataloger som inte ska finnas med i listan i Source Viewer. Sökvägen är relativ till programkatalogen. Jokertecken får inte användas.

I exempelvis följande JavaScript-kod öppnas Source Viewer-fönstret med alla källfiler med undantag för filen AIRSourceViewer.js och filerna i underkatalogerna Images och Sounds:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.exclude = ["AIRSourceViewer.js", "Images" "Sounds"];
viewer.viewSource(configObj);
```

initialPosition
En matris som innehåller två tal för de inledande x- och y-koordinaterna för Source Viewer-fönstret.

I följande JavaScript-exempel öppnas Source Viewer-fönstret vid koordinaterna [40, 60] (X = 40, Y = 60):

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.initialPosition = [40, 60];
viewer.viewSource(configObj);
```

modal
Ett booleskt värde som anger om Source Viewer ska vara ett modalt ("true") eller ett icke-modalt ("false") fönster. Standard är att Source Viewer-fönstret är modalt.

I följande JavaScript-exempel öppnas Source Viewer-fönstret så att användaren kan använda både Source Viewer-fönstret och andra programfönster:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.modal = false;
viewer.viewSource(configObj);
```

typesToAdd
En matris med strängar som används för att ange filtyper som ska ingå i Source Viewer-listan, förutom de standardtyper som ingår.

Standard är att följande filtyper finns i Source Viewer:

- Textfiler – TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES och CONFIG
- Bildfiler – JPG, JPEG, PNG och GIF

  Om inget värde anges ingår alla standardtyper, förutom de som anges i egenskapen `typesToExclude`.

  I exempelvis följande JavaScript-kod öppnas Source Viewer-fönstret med VCF- och VCARD-filer:

  ```
  var viewer = air.SourceViewer.getDefault();
  var configObj = {};
  configObj.typesToAdd = ["text.vcf", "text.vcard"];
  viewer.viewSource(configObj);
  ```

  För varje filtyp som du anger måste du även ange "text" (för textfiler) eller "image" (för bildfiler).

typesToExclude
En matris med strängar som anger filtyperna som ska tas bort från Source Viewer.

Standard är att följande filtyper finns i Source Viewer:

- Textfiler – TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES och CONFIG
- Bildfiler – JPG, JPEG, PNG och GIF

  I exempelvis följande JavaScript-kod ska Source Viewer-fönstret öppnas utan GIF- eller XML-filer:

  ```
  var viewer = air.SourceViewer.getDefault();
  var configObj = {};
  configObj.typesToExclude = ["image.gif", "text.xml"];
  viewer.viewSource(configObj);
  ```

  För varje filtyp som du anger måste du även ange `"text"` (för textfiler) eller `"image"` (för bildfiler).

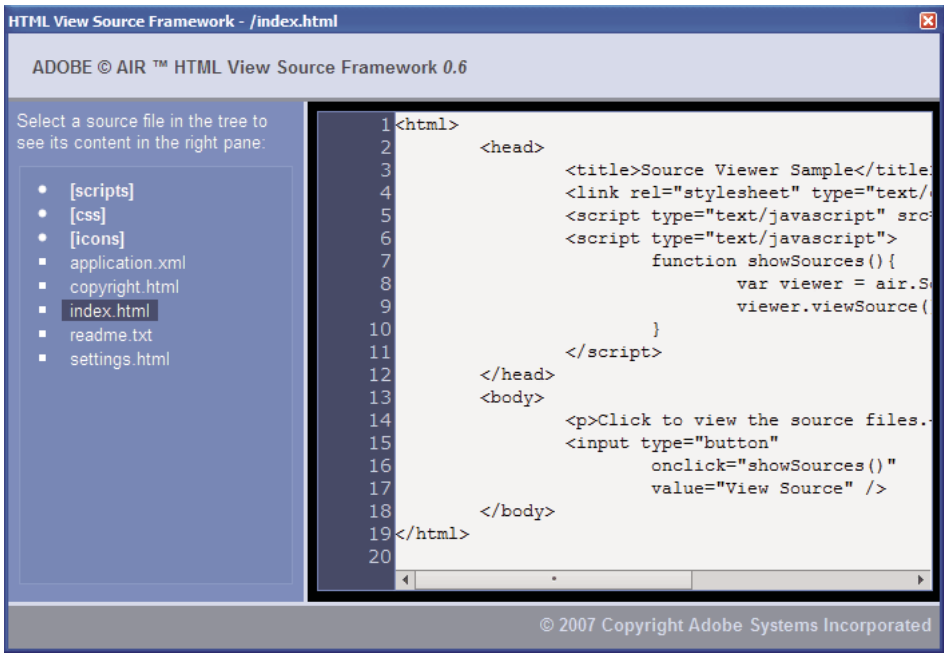
Öppna Source Viewer

Du ska även ta med element för användargränssnittet, exempelvis länkar, knappar eller menykommandon, som anropar Source Viewer-koden när användaren använder elementet. I följande exempel öppnas i Source Viewer ett enkelt program när användaren klickar på en länk:

```
<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>
```

Användargränssnitt i Source Viewer

När programmet anropar metoden `viewSource()` för ett SourceViewer-objekt, öppnas i AIR-programmet ett Source Viewer-fönster. I fönstret finns en lista med källfiler och kataloger (till vänster) och ett visningsområde där källkoden för den markerade filen visas (till höger):



Kataloger visas inom hakparenteser. Användaren kan klicka på en parentes för att expandera eller minimera kataloglistan.

I Source Viewer går det att visa källan för textfiler med godkända filtillägg (till exempel HTML, JS, TXT och XML) eller för bildfiler med godkända filtillägg (JPG, JPEG, PNG och GIF). Ett felmeddelande visas om användaren markerar en fil med ett otillåtet filtillägg ["Cannot retrieve text content from this filetype" ("Det går inte att hämta textinnehåll från filer med den här filtypen")].

De källfiler som exkluderades med metoden `setup()` visas inte (se ”Läsa in, konfigurera och öppna Source Viewer” på sidan 273).

Kapitel 19: Felsökning med AIR HTML Introspector

I Adobe® AIR® SDK finns JavaScript-filen AIRIntrospector.js som du kan ta med i ditt program för att få hjälp med att felsöka HTML-baserade program.

Om AIR Introspector

Med Adobe AIR HTML/JavaScript Application Introspector (även kallad AIR HTML Introspector) får du tillgång till kraftfulla funktioner för att utveckla och felsöka HTML-baserade program:

- Den innehåller ett undersökningsverktyg som gör att du kan peka på ett användargränssnittselement i programmet och se dess kod och DOM-egenskaper.
- Den innehåller en konsol för att skicka objektreferenser för undersökning och du kan ändra egenskapsvärden och köra JavaScript-kod. Du kan dessutom serialisera objekt till konsolen, vilket hindrar dig från att redigera data. Du kan även kopiera och spara text från konsolen.
- Den innehåller ett trädvy med DOM-egenskaper och -funktioner.
- Den gör det möjligt för dig att redigera attribut och textnoder för DOM-element.
- Här listas länkar, CSS-format, bilder och JavaScript-filer inlästa i programmet.
- Den gör att du kan se den ursprungliga HTML-källan och den aktuella kodkällan för användargränssnittet.
- Den ger dig tillgång till filer i programkatalogen. (Den här funktionen är endast tillgänglig för AIR HTML Introspector-konsolen som är öppen för programsandlådan. Den är inte tillgänglig för konsolerna som är öppna för innehåll som inte tillhör programsandlådan.)
- Den innehåller ett visningsprogram för XMLHttpRequest-objekt och deras egenskaper, inklusive egenskaperna `responseText` och `responseXML` (när de är tillgängliga).
- Du kan söka efter matchande text i källkoden och filerna.

Läsa in kod för AIR Introspector

Koden för AIR Introspector finns i JavaScript-filen, AIRIntrospector.js, som ingår i ramverkskatalogen för AIR SDK. Om du vill använda AIR Introspector i ditt program ska du kopiera AIRIntrospector.js till programmets projektkatalog och läsa in filen via en script-tagg i HTML-huvudfilen i programmet:

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

Du ska även inkludera filen i alla HTML-filer som motsvarar olika inbyggda fönster i ditt program.

Viktigt! *Inkludera AIRIntrospector.js-filen endast medan du utvecklar eller felsöker programmet. Ta bort den från det paketerade AIR-programmet som du tänker distribuera.*

I AIRIntrospector.js-filen definieras en klass, Console, som du kommer åt från JavaScript-koden genom att anropa `air.Introspector.Console`.

Obs! *Kod där AIR Introspector används måste finnas i programmets säkerhetssandlåda (i en fil i programkatalogen).*

Inspektera ett objekt på fliken Konsol

I Console-klassen definieras fem metoder: `log()`, `warn()`, `info()`, `error()` och `dump()`.

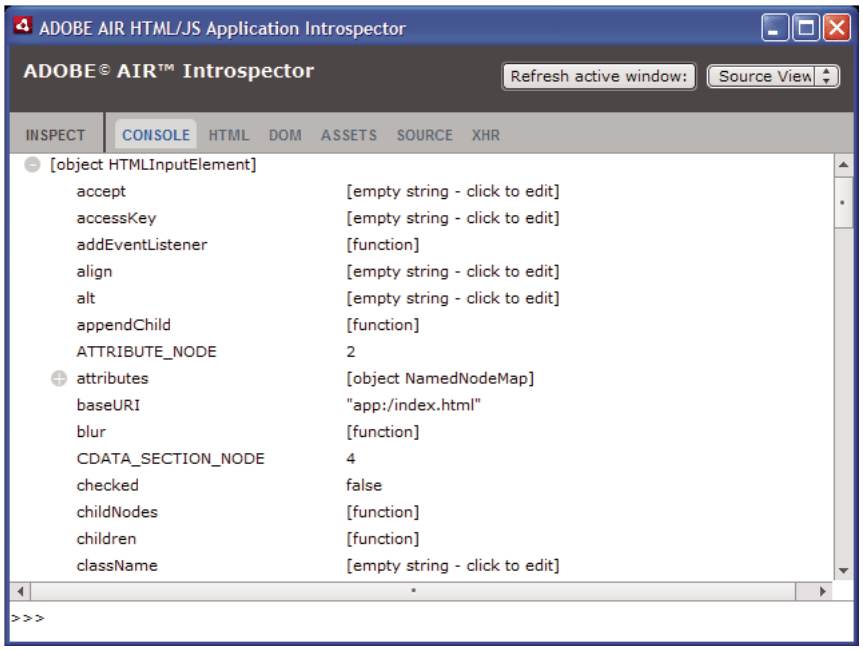
Med metoderna `log()`, `warn()`, `info()` och `error()` kan du skicka ett objekt till fliken Konsol. Den mest enkla av dessa metoder är `log()`-metoden. I följande kod skickas ett enkelt objekt, presenterat av variabeln `test`, till fliken Konsol:

```
var test = "hello";
air.Introspector.Console.log(test);
```

Det är emellertid mer användbart att skicka sammansatta objekt till fliken Konsol. På exempelvis följande HTML-sida finns en knapp (`btn1`) som anropar en funktion som skickar själva knappobjektet till fliken Konsol:

```
<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>
```

När du klickar på knappen visas btn1-objektet på fliken Konsol. Du kan expandera trädvyn för objektet om du vill kontrollera dess egenskaper:



Du kan redigera en objektsegenskap genom att i listan till höger klicka på egenskapsnamnet och ändra textlistan.

Metoderna `info()`, `error()` och `warn()` fungerar som `log()`-metoden. Däremot när du anropar dessa metoder visas på konsolen en ikon i början av raden:

Metod	Ikon
`info()`	
`error()`	
`warn()`	

Med metoderna `log()`, `warn()`, `info()` och `error()` skickas en referens endast till det faktiska objektet, vilket innebär att de egenskaper som är tillgängliga är de som du för närvarande ser. Om du vill serialisera det faktiska objektet ska du använda metoden `dump()`. Den här metoden har två parametrar:

Parameter	Beskrivning
`dumpObject`	Objektet som ska serialiseras.
`levels`	Det högsta antalet nivåer som ska undersökas i objektträdet (förutom rotninvån). Standardvärdet är 1, vilket innebär att en nivå ovanför rotnivån på trädet visas. Den här parametern är valfri.

När metoden `dump()` anropas serialiseras ett objekt innan det skickas till fliken Konsol, vilket medför att du inte kan redigera objektsegenskaperna. Titta på följande kod:

```
var testObject = new Object();
testObject.foo = "foo";
testObject.bar = 234;
air.Introspector.Console.dump(testObject);
```

När du kör den här koden visas `testObject`-objektet och dess egenskaper på konsolen, men du kan inte redigera egenskapsvärdena på konsolen.

Konfigurera AIR Introspector

Du konfigurerar konsolen genom att ställa in egenskaper för den globala variabeln `AIRIntrospectorConfig`. I exempelvis följande JavaScript-kod konfigureras AIR Introspector så att kolumner bryts vid 100 tecken:

```
var AIRIntrospectorConfig = new Object();
AIRIntrospectorConfig.wrapColumns = 100;
```

Se till att du ställer in egenskaperna för variabeln `AIRIntrospectorConfig` innan du läser in AIRIntrospector.js-filen (med en `script`-tagg).

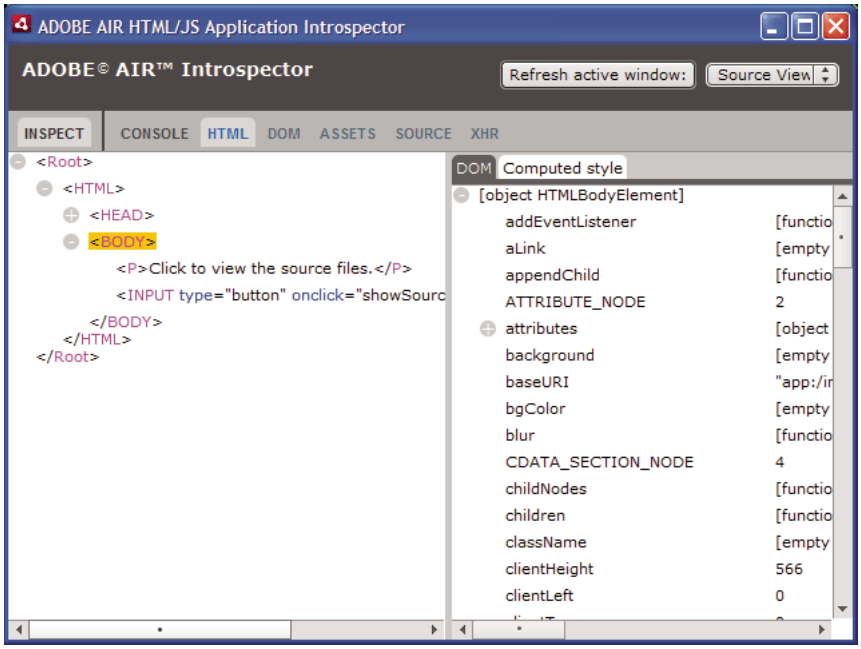
Det finns åtta egenskaper för variabeln `AIRIntrospectorConfig`:

Egenskap	Standardvärde	Beskrivning
`closeIntrospectorOnExit`	`true`	Ställer in Inspector-fönstret så att det stängs när alla andra fönster i applikationen är stängda.
`debuggerKey`	123 (F12-tangenten)	Nyckelkoden för kortkommandot för att visa och dölja fönstret AIR Introspector.
`debugRuntimeObjects`	`true`	Ställer in Introspector för att expandera körningsobjekt, förutom objekt definierade i JavaScript.
`flashTabLabels`	`true`	Ställer in konsolen och XMLHttpRequest-flikarna så att de börjar blinka, vilket visar att en förändring inträffat i dem (till exempel när text loggas på dessa flikar).
`introspectorKey`	122 (F11-tangenten)	Nyckelkoden för kortkommandon som används för att öppna panelen Inspektera.
`showTimestamp`	`true`	Ställer in fliken Konsol så att tidsstämplar visas i början av varje rad.
`showSender`	`true`	Ställer in fliken Konsol så att information visas i början av varje rad för objektet som skickar meddelandet.
`wrapColumns`	2000	Antalet kolumner där källfiler kommer att brytas.

Gränssnittet för AIR Introspector

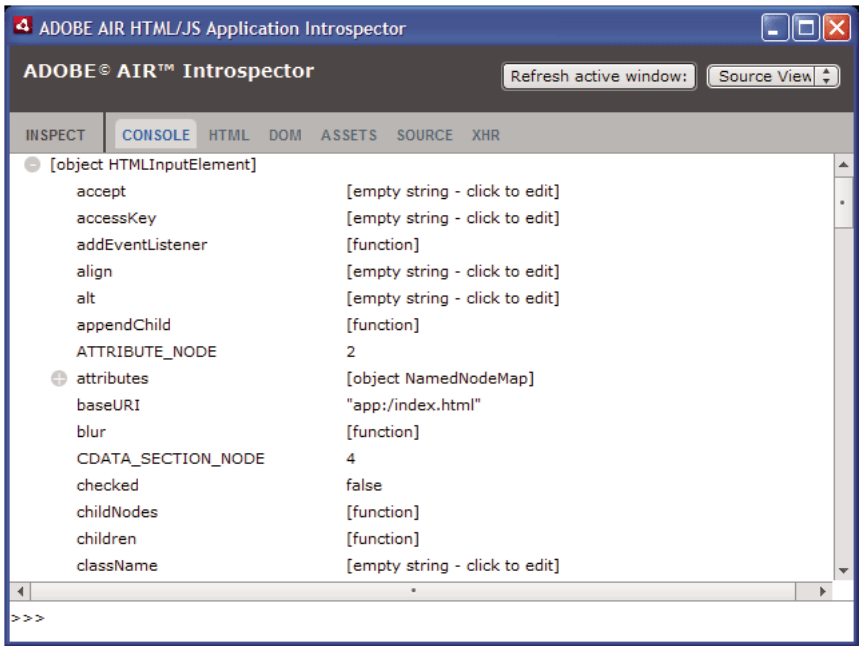
Tryck på tangenten F12 eller anropa en av metoderna i Console-klassen när du vill öppna fönstret AIR Introspector för att felsöka programmet (se ”Inspektera ett objekt på fliken Konsol” på sidan 278). Du kan konfigurera snabbtangenten så att du kan använda en annan tangent än F12-tangenten; se ”Konfigurera AIR Introspector” på sidan 280.

Fönstret AIR Introspector innehåller sex flikar – Konsol, HTML, DOM, Resurser, Källa och XHR – vilket framgår av bilden nedan:



Fliken Konsol

På fliken Konsol visas värden för egenskaperna som skickas som parametrar till en av metoderna i air.Introspector.Console-klassen. Mer information finns i ”Inspektera ett objekt på fliken Konsol” på sidan 278.

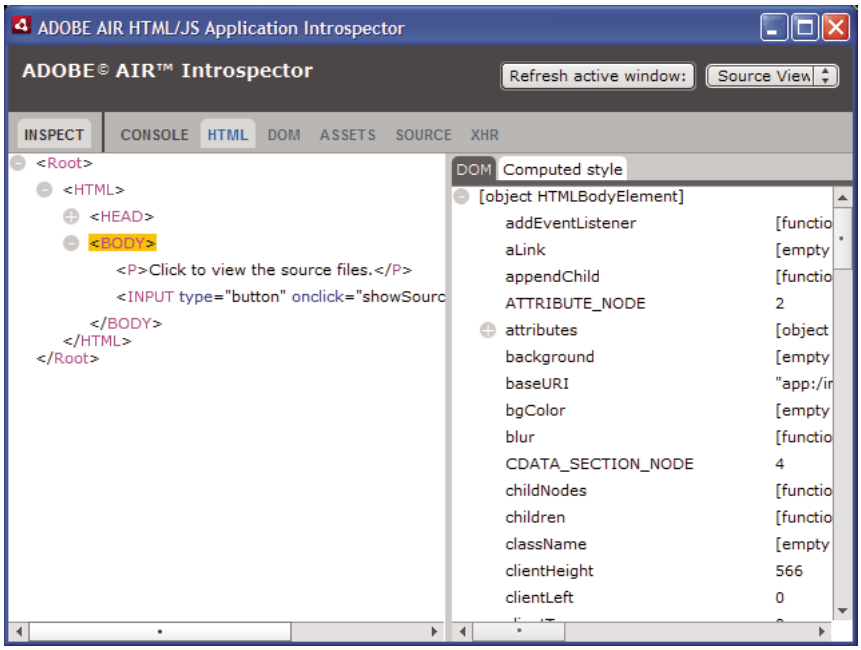


- Högerklicka på texten och välj Rensa konsol när du vill ta bort information i konsolen.
- Högerklicka på fliken Konsol och välj Spara konsol i fil när du vill spara texten på konsolfliken.

- Högerklicka på fliken Konsol och välj Spara konsol i urklipp när du vill spara texten på konsolfliken i urklippshanteraren. Högerklicka på texten och välj Kopiera när du vill kopiera den markerade texten.
- Högerklicka på fliken Konsol och välj Spara konsol i fil när du vill spara texten i Console-klassen.
- Tryck på CTRL+F i Windows eller Command+F i Mac OS när du vill söka efter text som finns på fliken. (Det går endast att söka i trädnoder som visas.)

Fliken HTML

På fliken HTML kan du visa hela HTML DOM i en trädstruktur. Klicka på ett element för att visa dess egenskaper till höger på fliken. Klicka på tecken + och - för att expandera respektive minimera en nod i trädet.



Du kan redigera alla attribut och textelement på HTML-fliken, och de redigerade värdena påverkar sedan programmet.

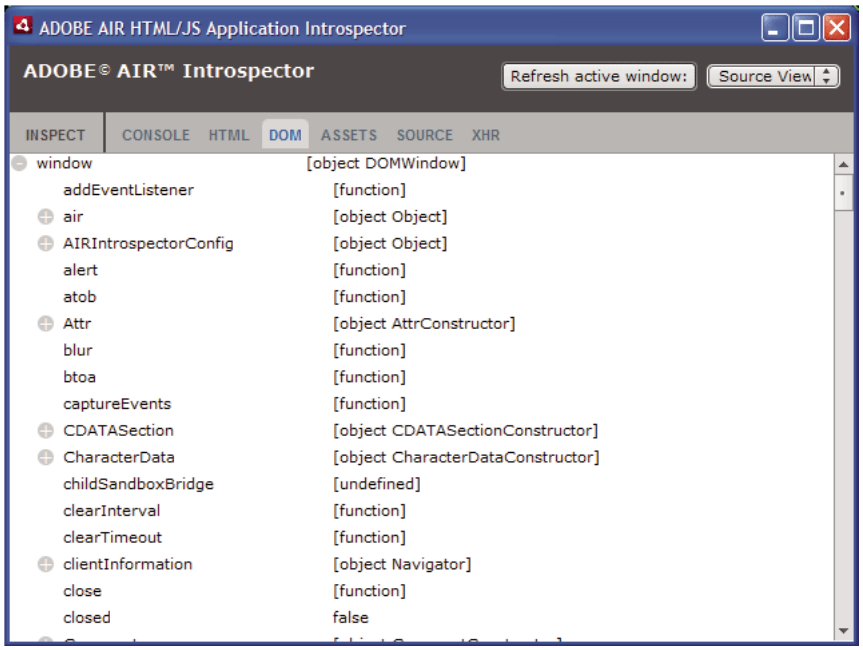
Klicka på knappen Inspektera (till vänster om fliklistan i AIR Introspector-fönstret). Du kan klicka på alla element på HTML-sidan i huvudfönstret och de associerade DOM-objekten kommer då att visas på fliken HTML. När huvudfönstret är aktivt kan du trycka på kortkommandot för att aktivera eller inaktivera knappen Inspektera. Kortkommandot är som standard F11. Du kan konfigurera kortkommando så att du kan använda en annan tangent än F11-tangenten; se ”Konfigurera AIR Introspector” på sidan 280.

Klicka på knappen Uppdatera aktivt fönster (högst upp i fönstret AIR Introspector) för att uppdatera de data som visas på fliken HTML.

Tryck på CTRL+F i Windows eller på Command+F i Mac OS för att söka efter text som visas på fliken. (Det går endast att söka i trädnoder som visas.)

Fliken DOM

På fliken DOM visas fönsterobjekten i en trädstruktur. Du kan redigera strängarna och de numeriska egenskaperna. De redigerade värdena visas i programmet.

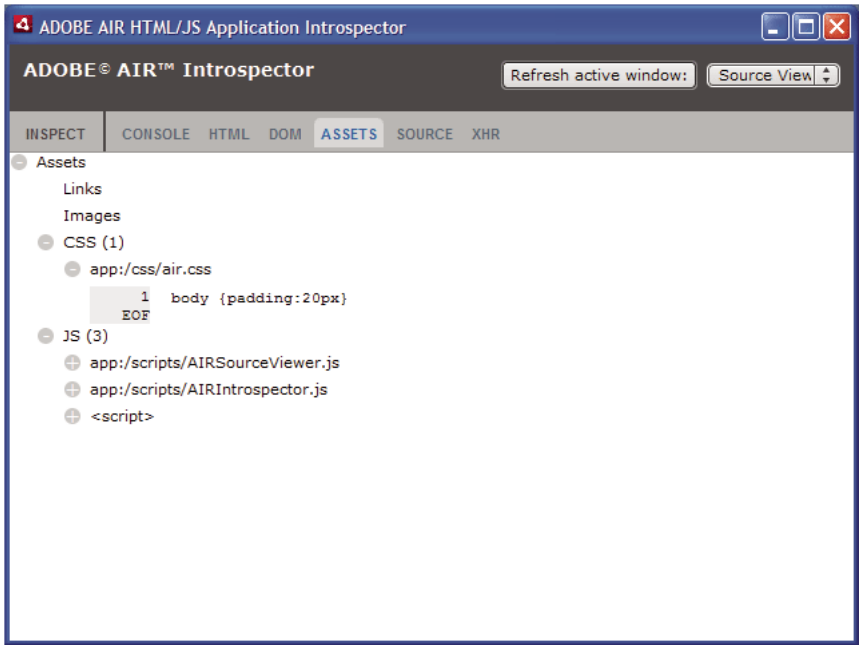


Klicka på knappen Uppdatera aktivt fönster (högst upp i fönstret AIR Introspector) för att uppdatera de data som visas på fliken DOM.

Tryck på CTRL+F i Windows eller på Command+F i Mac OS för att söka efter text som visas på fliken. (Det går endast att söka i trädnoder som visas.)

Fliken Resurser

På fliken Resurser kan du kontrollera länkarna, bilderna samt CSS- och JavaScript-filerna som lästs in i det inbyggda fönstret. När du expanderar en av dessa noder visas filens innehåll eller den bild som används.



Klicka på knappen Uppdatera aktivt fönster (högst upp i fönstret AIR Introspector) för att uppdatera de data som visas på fliken Resurser.

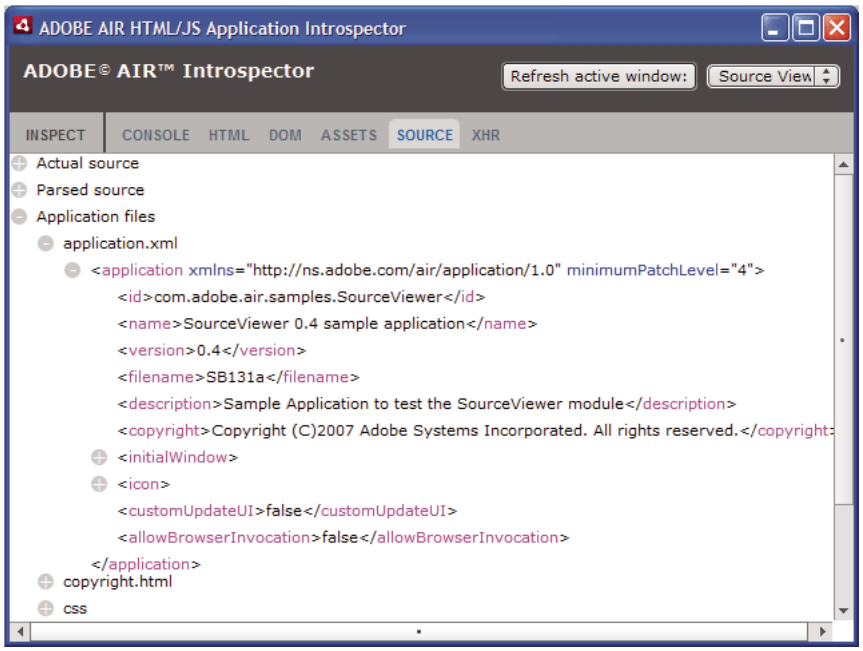
Tryck på CTRL+F i Windows eller på Command+F i Mac OS för att söka efter text som visas på fliken. (Det går endast att söka i trädnoder som visas.)

Fliken Källa

Fliken Källa innehåller tre avsnitt:

- Faktisk källa – Här visas sidans HTML-källa som lästes in i rotinnehållet när programmet startades.
- Tolkad källa – Här visas den aktuella koden som används för att skapa programmets gränssnitt, vilken kan skilja sig från den faktiska källan eftersom programmet automatiskt genererar kod med Ajax-tekniker.

- Programfiler – Här listas filerna i programkatalogen. Den här listan är bara tillgänglig för AIR Introspector när programmet startas från innehåll i programmets säkerhetssandlåda. I detta avsnitt kan du visa innehållet i textfiler och visa bilder.

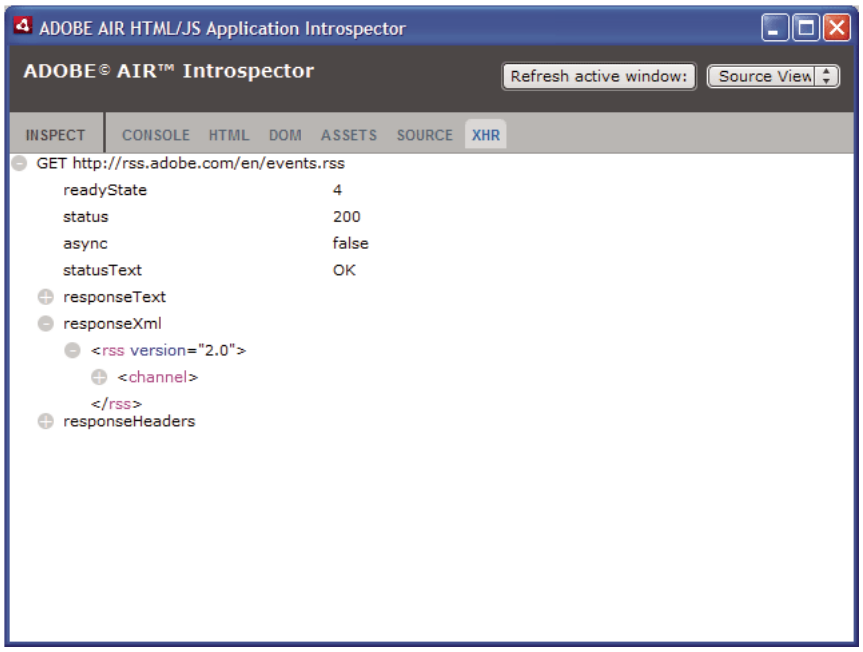


Klicka på knappen Uppdatera aktivt fönster (högst upp i fönstret AIR Introspector) för att uppdatera de data som visas på fliken Källa.

Tryck på CTRL+F i Windows eller på Command+F i Mac OS för att söka efter text som visas på fliken. (Det går endast att söka i trädnoder som visas.)

Fliken XHR

På fliken XHR fångas all XMLHttpRequest-kommunikation i programmet och informationen loggas. Detta medför att du kan se XMLHttpRequest-egenskaperna inklusive `responseText` och `responseXML` (när de är tillgängliga) i en trädvy.



Tryck på CTRL+F i Windows eller på Command+F i Mac OS för att söka efter text som visas på fliken. (Det går endast att söka i trädnoder som visas.)

Använda AIR Introspector med innehåll i en icke-programsandlåda

Du kan läsa in innehållet från programkatalogen i en iframe eller frame som är mappad till en icke-programsandlåda. (Läs HTML-säkerhet i Adobe AIR för ActionScript-utvecklare eller HTML security in Adobe AIR för HTML-utvecklare). Du kan använda AIR Introspector med sådant innehåll, men tänk på följande regler:

- Filen AIRIntrospector.js måste finnas med i innehållet både i programsandlådan och i icke-programsandlådan (iframe).
- Skriv inte över egenskapen `parentSandboxBridge` eftersom den används i AIR Introspector-koden. Lägg till egenskaper efter behov. Så i stället för att skriva följande:

  ```
  parentSandboxBridge = mytrace: function(str) {runtime.trace(str)}} ;
  ```

  ska du använda följande syntax:

  ```
  parentSandboxBridge.mytrace = function(str) {runtime.trace(str)};
  ```

- Från innehållet i icke-programsandlådan kan du inte öppna AIR Introspector genom att trycka på tangenten F12 eller genom att anropa en av metoderna i air.Introspector.Console-klassen. Du kan endast öppna Introspector-fönstret genom att klicka på knappen Introspector. Knappen läggs automatiskt till i det övre högra hörnet i iframe eller frame. (På grund av de säkerhetsrestriktioner som gäller för innehåll i icke-programsandlådan, kan ett nytt fönster endast öppnas som ett resultat från en åtgärd från användaren, till exempel när någon klickar på en knapp.)
- Du kan öppna olika AIR Introspector-fönster för programsandlådan och för icke-programsandlådan. Du kan skilja de båda åt genom att använda rubriken som visas i AIR Introspector-fönstret.
- På fliken Källa visas inte programfiler när AIR Introspector körs från en icke-programsandlåda.
- I AIR Introspector går det endast att se kod i sandlådan varifrån den öppnades.

Kapitel 20: Lokalisera AIR-program

Adobe AIR 1.1 och senare

I Adobe® AIR® finns stöd för många olika språk.

Om du vill ha en översikt av det lokaliserade innehållet i ActionScript 3.0 och Flex-ramverk ska du läsa "Lokalisera program" i Utvecklarhandbok för Adobe ActionScript 3.0.

Språk som stöds i AIR
Språkstöd för AIR-program i följande språk introducerades i versionen 1.1 av AIR:

- Förenklad kinesiska
- Traditionell kinesiska
- Franska
- Tyska
- Italienska
- Japanska
- Koreanska
- Portugisiska (Brasilien)
- Ryska
- Spanska

I version 1.5 av AIR lades följande språk till:

- Tjeckiska
- Nederländska
- Polska
- Svenska
- Turkiska

Fler hjälpavsnitt

Skapa flerspråkiga Flex-program i Adobe AIR

Skapa ett flerspråkigt HTML-baserat program

Lokalisera programnamnet och beskrivningen i AIR-programmets installationsprogram

Adobe AIR 1.1 och senare

Du kan ange flera språk för elementen `name` och `description` i programbeskrivningsfilen. Följande anger till exempel programnamnet på tre olika språk (engelska, franska och tyska):

```
<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>
```

`xml:lang`-attributet för varje text-element anger en språkkod enligt RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

name-elementet definierar det programnamn som AIR-programmets installationsprogram visar. AIR-programmets installationsprogram använder det lokaliserade värde som bäst överensstämmer med användargränssnittets språk som angetts i inställningarna för operativsystemet.

Du kan på liknande sätt ange flera språkversioner för `description`-elementet i programbeskrivningsfilen. Det här elementet definierar den beskrivningstext som AIR-programmets installationsprogram visar.

De här inställningarna gäller bara för de språk som är tillgängliga i AIR-programmets installationsprogram. De definierar inte de språkinställningar som finns för det installerade programmet som körs. AIR-program kan tillhandahålla användargränssnitt som har stöd för flera språk, förutom de som är tillgängliga i AIR-programmets installationsprogram.

Du hittar mer information i ”Element i AIR-programbeskrivningsfilen” på sidan 204.

Fler hjälpavsnitt

Skapa flerspråkiga Flex-program i Adobe AIR

Skapa ett flerspråkigt HTML-baserat program

Lokalisera HTML-innehåll med AIR HTML-lokaliseringsramverket

Adobe AIR 1.1 och senare

I AIR 1.1 SDK finns ett HTML-lokaliseringsramverk. JavaScript-filen AIRLocalizer.js definierar ramverket. Ramverkets katalog i AIR SDK innehåller filen AIRLocalizer.js. I den här filen finns en air.Localizer-klass som ger funktioner som hjälper till att skapa program som kan hantera flera lokaliserade versioner.

Läsa in koden för AIR:s HTML-lokaliseringsramverk

Om du vill använda lokaliseringsramverket kopierar du filen AIRLocalizer.js till ditt projekt. Inkludera den sedan i programmets HTML-huvudfil med hjälp av en script-tagg:

```
<script src="AIRLocalizer.js" type="text/javascript" charset="utf-8"></script>
```

Efterföljande JavaScript kan anropa `air.Localizer.localizer`-objektet:

```
<script>
    var localizer = air.Localizer.localizer;
</script>
```

`air.Localizer.localizer`-objektet är ett singleton-objekt som definierar metoder och egenskaper för användning och hantering av lokaliserade resurser. Klassen Localizer innehåller följande metoder:

Metod	Beskrivning
`getFile()`	Hämtar texten för ett angivet resurspaket för en angiven språkinställning. Se "Hämta resurser för en särskild språkinställning" på sidan 295.
`getLocaleChain()`	Returnerar språken i språkkedjan. Se "Definiera språkkedjan" på sidan 295.
`getResourceBundle()`	Returnerar paketnycklarna och motsvarande värden som ett objekt. Se "Hämta resurser för en särskild språkinställning" på sidan 295.
`getString()`	Hämtar den sträng som definierats för en resurs. Se "Hämta resurser för en särskild språkinställning" på sidan 295.
`setBundlesDirectory()`	Anger paketens katalogplats. Se "Anpassa inställningar för HTML-lokalisering i AIR" på sidan 294.
`setLocalAttributePrefix()`	Anger det prefix som används av de lokaliseringsattribut som används i HTML DOM-element. Se "Anpassa inställningar för HTML-lokalisering i AIR" på sidan 294
`setLocaleChain()`	Anger språkordningen i språkkedjan. Se "Definiera språkkedjan" på sidan 295.
`sortLanguagesByPreference()`	Sorterar språkinställningarna i språkkedjan utifrån ordningen för språkinställningar i operativsystemets inställningar. Se "Definiera språkkedjan" på sidan 295.
`update()`	Uppdaterar HTML DOM (eller ett DOM-element) med lokaliserade strängar från den aktuella språkkedjan. Mer information om språkkedjor finns i "Hantera språkkedjor" på sidan 291. Mer information om metoden `update()` finns i "Uppdatera DOM-element så att den aktuella språkinställningen används" på sidan 293.

Klassen Localizer innehåller följande statiska egenskaper:

Egenskap	Beskrivning
`localizer`	Returnerar en referens till det enkla Localizer-objektet för programmet.
`ultimateFallbackLocale`	Den språkinställning som används när programmet inte stöder användarinställningar. Se "Definiera språkkedjan" på sidan 295.

Ange språk som stöds

Använd elementet `<supportedLanguages>` i programbeskrivningsfilen för att identifiera de språk som programmet har stöd för. Det här elementet används bara av iOS, låsta Mac-miljöer och Android-program och ignoreras av alla andra programtyper.

Om du inte anger `<supportedLanguages>`-elementet utförs som standard följande åtgärder baserat på programtyp vid paketering:

- iOS – Alla språk som stöds av AIR-miljön listas i iOS App Store som språk som stöds för programmet.
- Låst Mac-miljö – Program som paketeras med ett låst miljöpaket har ingen lokaliseringsinformation.
- Android – Programpaketet har resurser för alla språk som stöds av AIR-miljön.

Du hittar mer information i "supportedLanguages" på sidan 234.

Definiera resurspaket

HTML-lokaliseringsramverket läser lokaliserade strängversioner från *lokaliserings* filer. En lokaliseringsfil är en samling nyckelbaserade värden som är serialiserade i en textfil. En lokaliseringsfil kallas ibland för ett *paket*.

Skapa en underkatalog för programmets projektkatalog med namnet locale. (Du kan även använda ett annat namn, se "Anpassa inställningar för HTML-lokalisering i AIR" på sidan 294.) I den här katalogen finns lokaliseringsfilerna. Katalogen kallas *paketkatalogen*.

För varje språkinställning som programmet kan hantera skapar du en underkatalog i paketkatalogen. Namnge varje underkatalog så att den överensstämmer med språkinställningskoden. Kalla till exempel den franska katalogen ”fr” och den engelska katalogen ”en”. Du kan använda ett understreck (_) om du vill definiera en språkinställning som har en språkkod och en landskod. Kalla till exempel katalogen för amerikansk engelska för ”en_us”. (Du kan också använda ett bindestreck i stället för ett understreck, t.ex. ”en-us”. HTML-lokaliseringsramverket kan hantera båda alternativen.)

Du kan lägga till valfritt antal resursfiler i en språkinställningsunderkatalog. Vanligtvis skapar du en lokaliseringsfil för varje språk (och placerar filen i katalogen för det språket). I HTML-lokaliseringsramverket finns en `getFile()`-metod som gör att du kan läsa innehåll i en fil (se ”Hämta resurser för en särskild språkinställning” på sidan 295.

Filer med filtillägget .properties kallas lokaliseringsegenskapsfiler. Du kan använda dem för att definiera nyckelvärdepar för en språkinställning. En egenskapsfil definierar ett strängvärde på varje rad. Följande definierar till exempel ett strängvärde `"Hello in English."` för en nyckel med namnet `greeting`:

```
greeting=Hello in English.
```

En egenskapsfil som innehåller följande text definierar sex nyckelvärdepar:

```
title=Sample Application
greeting=Hello in English.
exitMessage=Thank you for using the application.
color1=Red
color2=Green
color3=Blue
```

I det här exemplet visas en engelsk version av egenskapsfilen som ska sparas i katalogen en.

En fransk version av den här egenskapsfilen är placerad i katalogen fr:

```
title=Application Example
greeting=Bonjour en français.
exitMessage=Merci d'avoir utilisé cette application.
color1=Rouge
color2=Vert
color3=Bleu
```

Du kan definiera flera resursfiler för olika sorters information. Filen legal.properties kan till exempel innehålla formaterad, juridisk exempeltext (såsom upphovsrättsinformation). Du kan återanvända resursen i flera program. På liknande sätt kan du definiera separata filer som definierar lokaliserat innehåll för olika delar av användargränssnittet.

Använd UTF-8-kodning för de här filerna så att flera språk kan hanteras.

Hantera språkkedjor

När programmet läser in filen AIRLocalizer.js undersöks de språkinställningar som är definierade i programmet. De här språkinställningarna motsvarar underkatalogerna i paketkatalogen (se ”Definiera resurspaket” på sidan 290). Den här listan med tillgängliga språkinställningar kallas *språkkedja*. Filen AIRLocalizer.js sorterar språkkedjan automatiskt utifrån den önskade ordningen som angetts i operativsystemets inställningar. (`Capabilities.languages`-egenskapen listar operativsystemets användargränssnittsspråk i önskad ordning.)

Om ett program definierar resurser för språkinställningarna ”en”, ”en_US” och ”en_UK” så sorterar AIR:s HTML-lokaliseringsramverk språkkedjan på rätt sätt. Om ett program startas i ett system som har ”en” som primär språkinställning sorteras språkkedjan så här `["en", "en_US", "en_UK"]`. I det här fallet letar programmet först efter resurser i ”en”-paketet och sedan i ”en_US”-paketet.

Om systemet anger ”en-US” som primär språkinställning används dock sorteringsordningen `["en_US", "en", en_UK"]`. I det här fallet letar programmet först efter resurser i ”en_US”-paketet och sedan i ”en”-paketet.

Som standard definierar programmet den första språkinställningen i språkkedjan som den inställning som ska användas. Du kan be användaren välja en språkinställning första gången som programmet körs. Du kan sedan välja att spara valet i en inställningsfil och använda den språkinställningen vid efterföljande programstarter.

Programmet kan använda resurssträngar på alla språk i språkkedjan. Om en särskild språkinställning inte definierar en resurssträng använder programmet nästa matchande resurssträng för andra språk som är definierade i språkkedjan.

Du kan anpassa språkkedjan genom att anropa metoden `setLocaleChain()` för Localizer-objektet. Se ”Definiera språkkedjan” på sidan 295.

Uppdatera DOM-element med lokaliserat innehåll

Ett element i programmet kan referera till ett nyckelvärde i en lokaliseringsegenskapsfil. `title`-elementet i följande exempel anger till exempel ett `local_innerHTML`-attribut. Lokaliseringsramverket använder det här attributet för att leta reda på ett lokaliserat värde. Som standard söker ramverket efter attributnamn som börjar med `"local_"`. Ramverket uppdaterar de attribut som har namn som matchar texten som följer efter `"local_"`. I det här fallet ställer ramverket in `innerHTML`-attributet för `title`-elementet. `innerHTML`-attributet använder det värde som definierats för `mainWindowTitle`-nyckeln i standardegenskapsfilen (default.properties):

```
<title local_innerHTML="default.mainWindowTitle"/>
```

Om den aktuella språkinställningen inte definierar något matchande värde söker lokaliseringsramverket igenom resten av språkkedjan. Det använder nästa språkinställning i språkkedjan som ett värde är definierat för.

I följande exempel använder texten (`innerHTML`-attribut) i `p`-elementet värdet för den `greeting`-nyckel som definierats i standardegenskapsfilen:

```
<p local_innerHTML="default.greeting" />
```

I följande exempel använder värdeattributet (och visad text) i `input`-elementet värdet för den `btnBlue`-nyckel som definierats i standardegenskapsfilen:

```
<input type="button" local_value="default.btnBlue" />
```

Om du vill uppdatera HTML DOM så att de strängar som definierats i den aktuella språkkedjan används, anropar du metoden `update()` för Localizer-objektet. Anrop av metoden `update()` gör att Localizer-objektet tolkar DOM och tillämpar förändringar där det finns lokaliseringsattribut (`"local_..."`):

```
air.Localizer.localizer.update();
```

Du kan definiera värden för både ett attribut (t.ex. ”innerHTML”) och dess motsvarande lokaliseringsattribut (t.ex. ”local_innerHTML”). I så fall skriver lokaliseringsramverket bara över attributvärdet om det hittar ett matchande värde i lokaliseringskedjan. Följande element definierar till exempel både `value`- och `local_value`-attributen:

```
<input type="text" value="Blue" local_value="default.btnBlue"/>
```

Du kan också uppdatera ett särskilt DOM-element. Se nästa avsnitt, ”Uppdatera DOM-element så att den aktuella språkinställningen används” på sidan 293.

Som standard använder AIR:s HTML-lokaliseringsramverk `"local_"` som prefix för attribut som definierar lokaliseringsinställningar för ett element. Som standard definierar till exempel ett `local_innerHTML`-attribut det paket och resursnamn som används för `innerHTML`-värdet i ett element. Som standard definierar också ett `local_value`-attribut det paket och resursnamn som används för `value`-attributet i ett element. Du kan konfigurera Localizer-objektet så att ett annat attributprefix än `"local_"` används. Se ”Anpassa inställningar för HTML-lokalisering i AIR” på sidan 294.

Uppdatera DOM-element så att den aktuella språkinställningen används

När Localizer-objektet uppdaterar HTML DOM gör det så att markerade element använder attributvärden som baseras på strängar som är definierade i den aktuella språkkedjan. Om du vill att HTML-lokaliseringen ska uppdatera HTML DOM anropar du metoden `update()` för `Localizer`-objektet:

```
air.Localizer.localizer.update();
```

Om du bara vill uppdatera ett särskilt DOM-element skickar du det som en parameter till metoden `update()`. Metoden `update()` har bara en parameter, `parentNode`, som är valfri. Om `parentNode`-parametern är angiven definierar den DOM-elementet så att lokalisering görs. Om du anropar metoden `update()` och anger en `parentNode`-parameter ställs lokaliserade värden in för alla underordnade element som anger lokaliseringsattribut.

Här följer ett exempel på ett `div`-element:

```
<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>
```

Om du vill uppdatera det här elementet så att det använder lokaliserade strängar som definierats i den aktuella språkkedjan använder du följande JavaScript-kod:

```
var divElement = window.document.getElementById("colorsDiv");
air.Localizer.localizer.update(divElement);
```

Om ett nyckelvärde inte hittas i språkkedjan ställer lokaliseringsramverket in attributvärdet som värdet för `"local_"`-attributet. Anta till exempel att lokaliseringsramverket i föregående exempel inte hittar något värde för `lblColors`-nyckeln (i någon av default.properties-filerna i språkkedjan). I så fall använder det `"default.lblColors"` som `innerHTML`-värde. Om det här värdet används indikerar det (för utvecklaren) att resurser saknas.

Metoden `update()` skickar en `resourceNotFound`-händelse om den inte kan hitta en resurs i språkkedjan. Konstanten `air.Localizer.RESOURCE_NOT_FOUND` definierar strängen `"resourceNotFound"`. Händelsen har tre egenskaper: `bundleName`, `resourceName` och `locale`. `bundleName`-egenskapen är namnet på det paket där resursen inte hittas. `resourceName`-egenskapen är namnet på det paket där resursen inte hittas. `locale`-egenskapen är namnet på den språkinställning där resursen inte hittas.

Metoden `update()` skickar en `bundleNotFound`-händelse när den inte kan hitta det angivna paketet. Konstanten `air.Localizer.BUNDLE_NOT_FOUND` definierar strängen `"bundleNotFound"`. Händelsen har två egenskaper: `bundleName` och `locale`. `bundleName`-egenskapen är namnet på det paket där resursen inte hittas. `locale`-egenskapen är namnet på den språkinställning där resursen inte hittas.

Metoden `update()` utförs asynkront (och skickar händelserna `resourceNotFound` och `bundleNotFound` asynkront). Följande kod anger händelseavlyssnare för händelserna `resourceNotFound` och `bundleNotFound`:

```
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);
}
```

Anpassa inställningar för HTML-lokalisering i AIR

Med hjälp av metoden `setBundlesDirectory()` i Localizer-objektet kan du anpassa sökvägen till paketkatalogen. Metoden `setLocalAttributePrefix()` i Localizer-objektet gör att du kan anpassa sökvägen till paketkatalogen och anpassa det attributvärde som används av Localizer-objektet.

Standardpaketkatalogen definieras som språkinställningsunderkatalog för programkatalogen. Du kan ange en annan katalog genom att anropa metoden `setBundlesDirectory()` för Localizer-objektet. Den här metoden används med en parameter som sträng, `path`, som är sökvägen till den önskade paketkatalogen. Värdet för `path`-parametern kan vara något av följande:

- En sträng som definierar en sökväg som är relativ till programkatalogen, t.ex. `"locales"`
- En sträng som definierar en giltig URL som använder URL-schemat `app`, `app-storage` eller `file`, t.ex. `"app://languages"` (använd *inte* URL-schemat `http`)
- Ett File-objekt

Mer information om URL-adresser och katalogsökväger finns i:

- Sökvägar till File-objekt (för ActionScript-utvecklare)
- Paths of File objects (för HTML-utvecklare)

Följande kod anger till exempel paketkatalogen till en språkunderkatalog för programlagringskatalogen (inte programkatalogen):

```
air.Localizer.localizer.setBundlesDirectory("languages");
```

Skicka en giltig sökväg som `path`-parametern. Annars orsakar metoden ett BundlePathNotFoundError-undantag. Det här felet har `"BundlePathNotFoundError"` som `name`-egenskap och dess `message`-egenskap anger den ogiltiga sökvägen.

Som standard använder AIR:s HTML-lokaliseringsramverk `"local_"` som prefix för attribut som definierar lokaliseringsinställningar för ett element. `local_innerHTML`-attributet definierar till exempel det paket och resursnamn som används för `innerHTML`-värdet i följande `input`-element:

```
<p local_innerHTML="default.greeting" />
```

Metoden `setLocalAttributePrefix()` i Localizer-objektet gör att du kan använda ett annat attributprefix än `"local_"`. Den här statiska metoden använder en parameter som är den sträng som du vill använda som attributprefix. Följande kod ställer till exempel in så att lokaliseringsramverket använder ”loc_” som attributprefix:

```
air.Localizer.localizer.setLocalAttributePrefix("loc_");
```

Du kan anpassa det attributprefix som lokaliseringsramverket använder. Du vill kanske anpassa prefixet om standardvärdet (`"local_"`) hamnar i konflikt med namnet på ett annat attribut som används av din kod. Var noga med att använda giltiga tecken för HTML-attribut när du anropar den här metoden. (Värdet kan t.ex. inte innehålla ett blanksteg.)

Mer information om hur du använder lokaliseringsattribut i HTML-element finns i ”Uppdatera DOM-element med lokaliserat innehåll” på sidan 292.

Inställningarna för paketkatalogen och attributprefix finns inte kvar vid olika programsessioner. Om du använder en anpassad inställning för paketkatalog eller attributprefix måste du ange den varje gång som programmet startas.

Definiera språkkedjan

Som standard ställs språkkedjan in när du läser in AIRLocalizer.js-koden. De språkinställningar som finns i paketkatalogen och operativsystemets språkinställningar definierar den här språkkedjan. (Mer information finns i ”Hantera språkkedjor” på sidan 291.)

Du kan ändra språkkedjan genom att anropa metoden `setLocaleChain()` för Localizer-objektet. Du vill kanske anropa den här metoden om användaren anger en inställning för ett särskilt språk. Metoden `setLocaleChain()` använder en parameter, `chain`, som är en array med språkinställningar, t.ex. `["fr_FR","fr","fr_CA"]`. Språkinställningarnas ordning i arrayen anger den ordning i vilken ramverket söker efter resurser (i efterföljande åtgärder). Om en resurs inte hittas för den första språkinställningen i kedjan fortsätter ramverket att leta i andra språkinställningars resurser. Om `chain`-argumentet saknas, inte är en array eller är en tom array, så misslyckas funktionen och orsakar ett IllegalArgumentsError-undantag.

Den statiska metoden `getLocaleChain()` i Localizer-objektet returnerar en array som innehåller de språkinställningar som finns i den aktuella språkkedjan.

Följande kod läser den aktuella språkkedjan och lägger till två franska språkinställningar längst upp i kedjan:

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
```

Metoden `setLocaleChain()` skickar en `"change"`-händelse när den uppdaterar språkkedjan. Konstanten `air.Localizer.LOCALE_CHANGE` definierar strängen `"change"`. Händelsen har en egenskap, `localeChain`, som är en array med språkinställningskoder i den nya språkkedjan. Följande kod anger en händelseavlyssnare för den här händelsen:

```
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);
}
```

Den statiska egenskapen `air.Localizer.ultimateFallbackLocale` representerar den språkinställning som används när programmet inte stöder användarinställningar. Standardvärdet är `"en"`. Du kan ange en annan språkinställning enligt följande kod:

```
air.Localizer.ultimateFallbackLocale = "fr";
```

Hämta resurser för en särskild språkinställning

Metoden `getString()` för Localizer-objektet returnerar den sträng som är definierad för en resurs i en särskild språkinställning. Du behöver inte ange ett `locale`-värde när du anropar metoden. I det här fallet ser metoden på hela språkkedjan och returnerar strängen i den första språkinställningen som innehåller angivet resursnamn. Metoden har följande parametrar:

Parameter	Beskrivning
`bundleName`	Paketet som innehåller resursen. Det här är filnamnet på properties-filen utan tillägget .properties. Om den här parametern till exempel är angiven som `"alerts"`, tittar Localizer-koden i de lokaliseringsfiler som har namnet alerts.properties.
`resourceName`	Resursnamnet.
`templateArgs`	Valfritt. En array med strängar som ersätter numrerade taggar i ersättningssträngen. Här följer ett exempel på hur du kan anropa en funktion där `templateArgs`-parametern är `["Raúl", "4"]` och den matchande resurssträngen är `"Hello, {0}. You have {1} new messages."`. I det här fallet returnerar funktionen `"Hello, Raúl. You have 4 new messages."`. Om du vill ignorera den här inställningen skickar du ett `null`-värde.
`locale`	Valfritt. Språkinställningskoden (t.ex. `"en"`, `"en_us"` eller `"fr"`) som ska användas. Om en språkinställning anges och inget matchande värde hittas, fortsätter inte metoden att söka efter värden i andra språkinställningar i språkkedjan. Om ingen språkinställningskod är angiven returnerar funktionen strängen i den första språkinställningen i språkkedjan som har ett värde för det angivna resursnamnet.

Lokaliseringsramverket kan uppdatera markerade HTML DOM-attribut. Du kan dock använda lokaliserade strängar på andra sätt. Du kan till exempel använda en sträng i dynamiskt genererad HTML eller som ett parametervärde i ett funktionsanrop. Följande kod anropar till exempel funktionen `alert()` med den sträng som är definierad i `error114`-resursen i standardegenskapsfilen för språkinställningen fr_FR:

```
alert(air.Localizer.localizer.getString("default", "error114", null, "fr_FR"));
```

Metoden `getString()` skickar en `resourceNotFound`-händelse när den inte kan hitta resursen i det angivna paketet. Konstanten `air.Localizer.RESOURCE_NOT_FOUND` definierar strängen `"resourceNotFound"`. Händelsen har tre egenskaper: `bundleName`, `resourceName` och `locale`. `bundleName`-egenskapen är namnet på det paket där resursen inte hittas. `resourceName`-egenskapen är namnet på det paket där resursen inte hittas. `locale`-egenskapen är namnet på den språkinställning där resursen inte hittas.

Metoden `getString()` skickar en `bundleNotFound`-händelse när den inte kan hitta det angivna paketet. Konstanten `air.Localizer.BUNDLE_NOT_FOUND` definierar strängen `"bundleNotFound"`. Händelsen har två egenskaper: `bundleName` och `locale`. `bundleName`-egenskapen är namnet på det paket där resursen inte hittas. `locale`-egenskapen är namnet på den språkinställning där resursen inte hittas.

Metoden `getString()` utförs asynkront (och skickar händelserna `resourceNotFound` och `bundleNotFound` asynkront). Följande kod anger händelseavlyssnare för händelserna `resourceNotFound` och `bundleNotFound`:

```
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-objektets `getResourceBundle()`-metod returnerar angivet paket för en viss språkinställning. Värdet som returneras av metoden är ett objekt med egenskaper som matchar nycklarna i paketet. (Om programmet inte hittar det angivna paketet returnerar metoden ett `null`-värde.)

Metoden har två parametrar, `locale` och `bundleName`.

Parameter	Beskrivning
`locale`	Språkinställningen (t.ex. `"fr"`).
`bundleName`	Paketnamnet.

Följande kod anropar exempelvis metoden `document.write()` för att läsa in standardpaketet för språkinställningen fr. Därefter anropar koden metoden `document.write()` för att skriva värden för `str1`- och `str2`-nycklarna i paketet:

```
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/>");
```

Metoden `getResourceBundle()` skickar en `bundleNotFound`-händelse om det angivna paketet inte kan hittas. Konstanten `air.Localizer.BUNDLE_NOT_FOUND` definierar strängen `"bundleNotFound"`. Händelsen har två egenskaper: `bundleName` och `locale`. `bundleName`-egenskapen är namnet på det paket där resursen inte hittas. `locale`-egenskapen är namnet på den språkinställning där resursen inte hittas.

Metoden `getFile()` i Localizer-objektet returnerar innehållet i ett paket som en sträng för en angiven språkinställning. Paketfilen blir läst som en UTF-8-fil. Metoden innehåller följande parametrar:

Parameter	Beskrivning
`resourceFileName`	Filnamnet på resursfilen (t.ex. `"about.html"`).

`templateArgs`	Valfritt. En array med strängar som ersätter numrerade taggar i ersättningssträngen. Här följer ett exempel på hur du kan anropa en funktion där `templateArgs`-parametern är `["Raúl", "4"]` och den matchande resursfilen innehåller två rader:
	`<html>`
	`<body>Hello, {0}. You have {1} new messages.</body>`
	`</html>`
	I det här fallet returnerar funktionen en sträng med två rader:
	`<html>`
	`<body>Hello, Raúl. You have 4 new messages. </body>`
	`</html>`

`locale`	Språkinställningskoden som ska användas, t.ex. `"en_GB"`. Om en språkinställning anges och ingen matchande fil hittas, fortsätter inte metoden att söka i andra språkinställningar i språkkedjan. Om *ingen* språkinställningskod är angiven returnerar funktionen den text i den första språkinställningen i språkkedjan som har en fil som matchar `resourceFileName`.

Följande kod anropar till exempel metoden `document.write()` med hjälp av innehållet i about.html-filen för språkinställningen fr:

```
var aboutWin = window.open();
var aboutHtml = localizer.getFile("about.html", null, "fr");
aboutWin.document.close();
aboutWin.document.write(aboutHtml);
```

Metoden `getFile()` skickar en `fileNotFound`-händelse när den inte kan hitta en resurs i språkkedjan. Konstanten `air.Localizer.FILE_NOT_FOUND` definierar strängen `"resourceNotFound"`. Metoden `getFile()` utförs asynkront (och skickar händelsen `fileNotFound` asynkront). Händelsen har två egenskaper: `fileName` och `locale`. Egenskapen `fileName` är namnet på den fil som inte hittas. `locale`-egenskapen är namnet på den språkinställning där resursen inte hittas. Följande kod anger en händelseavlyssnare för den här händelsen:

```
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);
}
```

Fler hjälpavsnitt

Skapa ett flerspråkigt HTML-baserat program

Kapitel 21: Systemvariabeln path

AIR SDK innehåller några program som kan startas från en kommandorad eller ett terminalfönster. Det är ofta enklare att köra dessa program om sökvägen till bin-katalogen för SDK ingår i systemvariabeln path.

Den information som finns här behandlar hur du anger sökvägen i Windows, Mac och Linux och kan användas som riktlinjer. Men eftersom olika datorkonfigurationer ofta skiljer sig åt markant kan det hända att proceduren inte fungerar på alla system. I så fall bör du hitta den information du behöver i dokumentationen till operativsystemet eller på nätet.

Ange PATH i Linux och Mac OS med Bash-kommandotolken

När du skriver ett kommando i ett terminalfönster måste kommandotolken (ett program som tolkar det du skriver och försöker reagera som förväntat) först hitta kommandoprogrammet i filsystemet. Kommandotolken söker efter kommandon i en lista med kataloger, som sparas i en systemvariabel med namnet $PATH. Om du vill visa vad som finns i sökvägen skriver du:

```
echo $PATH
```

Detta returnerar en kolonavgränsad lista över kataloger, som bör se ut ungefär så här:

```
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin
```

Målet är att lägga till sökvägen till bin-katalogen för AIR SDK i listan, så att kommandotolken kan hitta ADT- och ADL-verktygen. Förutsatt att du har placerat AIR SDK i `/Users/fred/SDKs/AIR` lägger följande kommando till de nödvändiga katalogerna i sökvägen:

```
export PATH=$PATH:/Users/fred/SDKs/AIR/bin:/Users/fred/SDKs/android/tools
```

Obs! *Om sökvägen innehåller tomma mellanslagstecken låter du dessa föregås av ett omvänt snedstreck, enligt följande:*

```
/Users/fred\ jones/SDKs/AIR\ 2.5\ SDK/bin
```

Du kan använda kommandot `echo` igen för att kontrollera att det fungerade:

```
echo $PATH
/usr/bin:/bin:/usr/sbin:/usr/local/bin:/usr/x11/bin:/Users/fred/SDKs/AIR/bin:/Users/fred/SDK
s/android/tools
```

Än så länge fungerar allt. Nu bör du kunna skriva följande kommandon och få det svar du önskar dig:

```
adt -version
```

Om du ändrade $PATH-variabeln korrekt bör kommandot rapportera ADT-versionen.

Ett problem kvarstår dock. Nästa gång du öppnar ett terminalfönster kommer du att märka att de nya posterna i sökvägen inte längre är kvar. Du måste köra kommandot för att ange sökvägen varje gång du öppnar ett nytt terminalfönster.

En vanlig lösning på det här problemet är att lägga till kommandot i ett av de startskript som används av kommandotolken. I Mac OS kan du skapa filen .bash_profile i katalogen ~/användarnamn, så körs den varje gång du öppnar ett nytt terminalfönster. I Ubuntu är det startskript som körs när du öppnar ett nytt terminalfönster .bashrc. Andra Linux-distributioner och -kommandotolkar har liknande konventioner.

Så här lägger du till kommandot i startskriptet för kommandotolken:

1. Gå till din hemkatalog:

   ```
   cd
   ```

2. Skapa konfigurationsprofilen för kommandotolken (vid behov) och dirigera om den text du skriver till slutet av filen med ”`cat >>`”. Använd lämplig fil för aktuellt operativsystem och aktuell kommandotolk. Du kan till exempel använda `.bash_profile` i Mac OS och `.bashrc` i Ubuntu.

   ```
   cat >> .bash_profile
   ```

3. Skriv den text som ska läggas till i filen:

   ```
   export PATH=$PATH:/Users/cward/SDKs/android/tools:/Users/cward/SDKs/AIR/bin
   ```

4. Avsluta textomdirigering genom att trycka på `CTRL-SHIFT-D` på tangentbordet.

5. Titta i filen för att försäkra dig om att allt är OK:

   ```
   cat .bash_profile
   ```

6. Öppna ett nytt terminalfönster för att kontrollera sökvägen:

   ```
   echo $PATH
   ```

   Dina sökvägstillägg bör nu vara med.

Om du senare skapar en ny version av en SDK i en annan katalog måste du uppdatera sökvägskommandot i konfigurationsfilen. Annars används den gamla versionen.

Ange sökväg i Windows

När du öppnar ett kommandofönster i Windows ärver det fönstret de globala systemvariabler som definierats i systemegenskaperna. En av de viktiga variablerna är path, den lista med kataloger som kommandotolken söker igenom när du skriver namnet på ett program som ska köras. Om du vill se vad som ingår i sökvägen när du använder ett kommandofönster skriver du:

```
set path
```

Då visas en semikolonavgränsad lista över kataloger, som ser ut ungefär så här:

```
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
```

Målet är att lägga till sökvägen till bin-katalogen för AIR SDK i listan, så att kommandotolken kan hitta ADT- och ADL-verktygen. Förutsatt att du har placerat AIR SDK i `/C:\SDKs\AIR` kan du lägga till rätt sökväg på följande sätt:

1. Öppna dialogrutan Systemegenskaper på Kontrollpanelen eller genom att högerklicka på ikonen Den här datorn och välja Egenskaper på menyn.

2. Klicka på knappen Miljövariabler på fliken Avancerat.

3. Välj posten Path i delen Systemvariabler i dialogrutan Miljövariabler.

4 Klicka på Redigera.

5 Rulla till slutet av texten i fältet Variabelvärde.

6 Skriv följande text allra sist i det aktuella värdet:

```
;C:\SDKs\AIR\bin
```

7 Klicka på OK i alla dialogrutor för att spara sökvägen.

Om du har kommandofönster öppna bör du vara medveten om att deras miljöer inte uppdateras. Öppna ett nytt kommandofönster och skriv följande kommando för att se till att sökvägarna är korrekt konfigurerade:

```
adt -version
```

Om du senare ändrar platsen för AIR SDK, eller lägger till en ny version, måste du uppdatera variabeln path igen.